Universidade Estadual de Campinas - UNICAMP
Faculdade de Engenharia Elétrica e de Computação - FEEC
Departamento de Engenharia de Computação e Automação
Industrial - DCA

Aplicação de Controlador Evolutivo a Pêndulo Servo Acionado

André Luiz Delai

Dissertação submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Elétrica

Banca Examinadora:
Prof. Dr. José Raimundo de Oliveira (Orientador) - DCA/FEEC/UNICAMP
Profa. Dra. Tatiane Jesus de Campos - ANHANGUERA EDUCACIONAL
Prof. Dr. Marconi Kolm Madrid - DSCE/FEEC/UNICAMP
Prof. Dr. Rafael Santos Mendes - DCA/FEEC/UNICAMP

Campinas, SP 25 de Janeiro de 2008

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Delai, André Luiz
D372a Aplicação de controlador evolutivo a pêndulo servo acionado / André Luiz Delai. --Campinas, SP: [s.n.], 2008.

Orientador: José Raimundo de Oliveira
Dissertação (Mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Sistema de controle digital. 2. Hardware. 3. Algoritmos evolutivos. 4. VHDL (Linguagem descritiva de hardware). 5. Dispositivos lógico programáveis. I. Oliveira, José Raimundo de. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Application of evolutionary controller to a pendulum driver
Palavras-chave em Inglês: Control systems, Evolutionary hardware, Reconfigurable hardware, FPGA, VHDL
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Tatiane de Jesus Campos, Marconi Kolm Madrid, Rafael Santos Mendes
Data da defesa: 25/01/2008
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: André Luiz Delai

Data da Defesa: 25 de janeiro de 2008

Título da Tese: "Aplicação de Controlador Evolutivo a Pêndulo Servo Acionado"

Prof. Dr. José Raimundo de Oliveira (Presidente): _____

Profa. Dra. Tatiane Jesus de Campos: _____

Prof. Dr. Marconi Kolm Madrid: _____

Prof. Dr. Rafael Santos Mendes: _____

Resumo

O uso de técnicas evolutivas empregando algorítmos genéticos na obtenção de projetos de circuitos eletrônicos analógicos e digitais já é fato e vêm sendo estudado a alguns anos. Neste contexto, o objetivo deste trabalho foi o de implementar em *hardware* reconfigurável a proposta de um controlador para pêndulo não-linear amortecido, obtido através de técnicas de *Hardware* Evolutivo. Para desenvolver um modelo físico baseado no modelo teórico (simulado) foram utilizadas tecnologias tais como a dos *Field Programable Gate Arrays* (FPGAs) e também a linguagem de descrição de *hardware VHSIC Hardware Description Language* (VHDL), dentre outros recursos.

Palavras-chave: Sistemas de Controle, *Hardware* Evolutivo, *Hardware* Reconfigurável, FPGA, VHDL.

Abstract

The use of genetic algorithms using evolutionary techniques in obtaining projects of analogue and digital electronic circuits is already fact and have been studied for some years. In this context, the objective of this work was the implementation in reconfigurable hardware of a driver for non-linear damped pendulum, obtained through Evolvable Hardware approach. Technologies such as the Field Programable Gate Arrays (FPGA's) and VHDL were used to develop a physical model based on the theoretical model(simulated), among other resources.

Keywords: Control Systems, Evolvable Hardware, Reconfigurable Hardware, FPGA, VHDL.

Agradecimentos

Agradeço às pessoas que de alguma forma contribuíram para a realização deste trabalho. Em especial, quero agradecer:

A meu orientador Prof. Dr. José Raimundo de Oliveira, por ter me aceitado como seu aluno e me guiado com paciência e competência durante todo o meu mestrado.

Ao Prof. Dr. Marconi Kolm Madrid por ter me disponibilizado acesso ao Laboratório de Sistemas Modulares Robóticos (LSMR) e pelo grande apoio moral.

Ao professor Dr. José Antenor Pomílio por ter gentilmente cedido um de seus módulos de potência, que foi usado no acionamento do motor do pêndulo.

A Tatiane Campos, pelas conversas, dicas e pelo excelente trabalho de doutorado o qual serviu de base para esta dissertação.

Aos professores da pós-graduação da FEEC, por toda a bagagem de conhecimento transmitida durante os cursos.

Ao meu pai Hélio, minha mãe Maria Lindaci e meu irmão Fábio, por terem acreditado em mim em todos os momentos e me ajudado a percorrer mais essa trajetória.

A minha avó Emília (*in memorian*), pelo amor infinito e dedicação dados a mim durante todos os momentos da minha vida.

Ao meu grande e verdadeiro amigo Filipe Fazanaro (exímio jogador de *Counter Strike Source*) que me apoiou, incentivou e me ajudou a concluir esse trabalho.

A todos os meus amigos, pela paciência e companheirismo.

A CAPES pelo apoio financeiro.

*A minha avó "Dona Bibi"(*in memorian*), infinitas saudades.*

"Existem comprimentos de ondas que pessoas não podem enxergar, sons que pessoas não podem ouvir, e talvez computadores tenham pensamentos que pessoas não poderiam ter."

Richard Hamming

Sumário

Lista de Abreviaturas

AG Algoritmo Genético

CC Corrente Contínua

E/S Entrada/Saida

EDA *Electronic Design Automation*

EHW *Evolvable Hardware*

FPGA *Field Programmable Gate Array*

HDL *Hardware Description Language*

IBM *International Business Machines*

IEEE *Institute of Electrical and Electronic Engineers*

IP *Internet Protocol*

JTAG *Joint Test Action Group*

LE *Logic Element*

LSMR Laboratório de Sistemas Modulares Robóticos

PC *Personal Computer*

PD Proporcional Derivativo

PI Proporcional Integral

PID Proporcional Integral Derivativo

PLD *Programable Logic Device*

PROM *Programable Read Only Memory*

PWM *Pulse Width Modulation*

SRAM *Static Random Access Memory*

TCP *Transmission Control Protocol*

ULA Unidade Lógica Aritmética

VHDL *VHSIC Hardware Description Language*

VHSIC *Very-High-Speed Integrated Circuit*

Mbps *Megabits* por segundo

Lista de Figuras

Lista de Tabelas

Capítulo 1

INTRODUÇÃO

A automação no processo de síntese de circuitos eletrônicos é empregada a muitos anos. Projetar circuitos eletrônicos, analógicos ou digitais, confiáveis, de baixo consumo e otimizados é, muitas vezes, uma tarefa difícil e que pode demandar um longo tempo de projeto. Em um mundo com fronteiras comerciais cada vez menores, tecnologia de ponta significa dinheiro e, novas técnicas de projeto que venham a somar ítens como produtividade, baixo custo e desempenho serão sempre bem vindas para a indústria da eletrônica.

Uma nova abordagem para o projeto automatizado de controladores foi proposta por Campos (2007), a qual realiza um processo heurístico buscando o melhor circuito eletrônico para uma determinada aplicação (naquele trabalho, um controlador digital para pêndulo). A busca é realizada através da técnica conhecida como *hardware* evolutivo, que emprega algoritmos evolutivos com o objetivo de evoluir circuitos até um estágio satisfatório a sua aplicação. A técnica EHW aplicada a controle de sistemas é uma alternativa aos métodos tradicionais, ela usa algoritmo genético no projeto e não para a sintonia de controladores. Nesta tese, os controladores evolutivos são comparados aos controladores clássicos Proporcionais e Proporcionais-Derivativos com ganhos sintonizados por meio de algoritmos genéticos (AG's), obtendo resultados promissores e inspirando a escolha do tema desta dissertação, que buscou validar a técnica através de um modelo experimental de bancada.

O desenvolvimento e testes dos controladores propostos por Campos (2007) foram realizados exclusivamente em ambiente computacional, sendo que os controladores foram simulados para análise de comportamento frente ao controle posicional de um pêndulo não-linear, e os dados obtidos na simulação utilizados como *feedback* no processo de evolução do *hardware*.

1.1 Objetivo Contribuição e Organização do Trabalho

O trabalho realizado como parte desta dissertação visou a implementação em laboratório do sistema simulado em computador por Campos (2007) para teste dos controladores evolutivos. O objetivo da construção física do sistema foi o de verificar o desempenho dos controladores através de testes em um pêndulo real. Ele contribuiu como um complemento daquele trabalho, pois colocou a prova o controlador evolutivo digital de 4 bits frente a um controlador proporcional também de 4 bits no controle de um pêndulo servo acionado, tendo-se imposto as condições de testes idênticas para ambos os controladores.

O texto está organizado em 5 capítulos, começando por este de introdução e os demais capítulos que são descritos a seguir. O capítulo 2 inicia com uma introdução aos conceitos fundamentais descrevendo um resumo sobre a área de computação evolutiva, a sua base na teoria neo-darwiniana e seu principal algoritmo evolutivo, o algoritmo genético (Holland, 1975), amplamente usado na pesquisa da evolução de circuitos eletrônicos. No âmbito dos algoritmos genéticos são detalhados os principais componentes dos algoritmos evolutivos como a codificação, avaliação e operadores de seleção e reprodução. Neste mesmo capítulo é apresentada a área de *hardware* evolutivo, que tem como base os conceitos abordados no capítulo anterior. São descritos os principais conceitos, benefícios e motivações desta área de pesquisa.

O capítulo 3 apresenta a tecnologia de dispositivos lógicos reconfiguráveis *Field Programmable Gate Array* (FPGA), que baseia o *chip* reconfigurável escolhido para ser utilizado na execução do projeto. Também é apresentada uma breve descrição da *VHSIC Hardware Description Language*(VHDL), utilizada na programação deste dispositivo.

Conceitos básicos e detalhes sobre o objeto do controle (pêndulo), controladores PID, a forma como o sistema atua sobre o pêndulo e também a técnica utilizada na geração do controlador evolutivo podem ser vistos no capítulo 4.

No capítulo 5 inicia-se com a descrição do trabalho propriamente dito onde é mostrada a teoria básica do sistema de um pêndulo e também a técnica utilizada na criação dos controladores. Na seqüencia são mostrados os recursos técnicos utilizados no desenvolvimento do sistema de controle do pêndulo em bancada, tal como a alimentação do sistema, circuito de interface e kit de desenvolvimento. Por fim são descritos de forma detalhada todos os módulos sintetizados dentro do *chip* digital reconfigurável, os quais são responsáveis pelo controle do sistema. Naquele capítulo também é vista a descrição do sistema de aquisição de dados utilizado para amostrar os sinais que tornaram possível a criação dos gráficos de análise comportamental dos controladores. São exibidos os gráficos obtidos assim como um cálculo comparativo de erro

entre os controladores evolutivo e proporcional sintonizado por AG.

O último capítulo contém as considerações finais sobre o trabalho, apresentando uma análise geral sobre os testes, a técnica de construção de controladores evolutivos e também uma discussão sobre trabalhos futuros.

Capítulo 2

COMPUTAÇÃO EVOLUTIVA E HARDWARE EVOLUTIVO

2.1 Computação Evolutiva

2.1.1 Resumo da Teoria da Seleção Natural

Em meados do século dezenove Charles Darwin apresentava à comunidade científica uma teoria muito polêmica que visava, em suma, explicar a diversidade biológica abundante em nosso planeta. A primeira edição do livro a Origem das Espécies esgotou-se do mercado no primeiro dia de publicação, 24 de novembro de 1859.

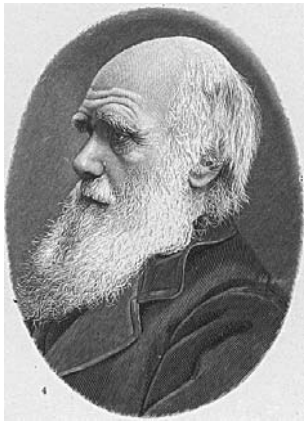

Figura 2.1: Charles Darwin

Darwin não foi o primeiro a propor a idéia de que as espécies de plantas e de animais

podem sofrer modificações com o passar do tempo. Na última década do séc. XVIII, o avô de Charles, Erasmus Darwin, escreveu um tratado sobre a evolução e logo depois, em 1809, o naturalista francês Jean Baptiste de Lamarck publicou sua Filosofia Zoológica, na qual apresenta estudos sobre a mutabilidade das espécies biológicas (Green, 2006). Entretanto, é de Darwin que se origina a moderna teoria da evolução.

Entre 1831 e 1836 Darwin realizou uma viagem à bordo do H. M. S. Beagle trabalhando como naturalista e colecionando e observando os diversos fenômenos geológicos e biológicos que encontrava. Até a apresentação de seu livro em 1859 realizou extensas leituras e fez experiências cuidadosas para fundamentar sua teoria.

Basicamente a teoria da seleção natural afirma que as espécies biológicas existentes hoje, ou em qualquer outra época, estão sujeitas à seleção por seu meio ambiente. Dessa forma, os indivíduos (seres biológicos) são forçados a se adaptar ao ambiente para terem melhores chances de sobrevivência e reprodução como, por exemplo, fugindo de predadores e conseguindo alimento. Os indivíduos com baixo nível de adaptação conseqüentemente perecem. Devido à variação, cujos resultados podem ser observados amplamente, a Seleção Natural pode, segundo Darwin, explicar a evolução biológica, ou o que ele chamava de "descendência com modificação".

De acordo com a teoria de Darwin, o chamado Darwinismo, para que haja a evolução, os indivíduos necessitam ser submetidos a três processos:

- Reprodução com herança
- Variação
- Seleção Natural

Na reprodução com herança os indivíduos transmitem aos seus descendentes características anteriormente premiadas pelo ambiente. Já na variação, mudanças aleatórias ocorrem para geração de novas características a serem testadas pela seleção natural. Dessa forma, o "motor evolucionário" reside na reprodução com variação e na seleção pelo meio ambiente.

Neodarwinismo

Darwin conhecia as alterações que ocorriam nos seres, e que algumas delas pelo menos eram hereditárias, porém não conhecia o mecanismo da hereditariedade devido ao fato de que, até então, não se haviam estabelecidos os conceitos da genética.

Gregor Mendel, monge austríaco, realizou por volta de 1866 experiências com plantas de horta, especialmente ervilhas, cujos resultados o levaram a propor a idéia de herança particulada. Mendel notou ao estudar algumas características como a cor das flores, a altura da planta, o formato e a textura da semente, que as contribuições parentais são expressas desigualmente a combinação não ocorre (Green, 2006).

Figura 2.2: Gregor Mendel

Para explicar seus resultados, Mendel sugeriu a idéia de que esses traços são herdados como elementos (que hoje chamamos de genes), sendo cada elemento recebido de um dos pais. Alguns elementos são dominantes, outros recessivos, de modo que se um descendente possui dois elementos dominantes, ou um dominante e um recessivo, o traço dominante é que será aparente. O recessivo só aparecerá quando estiverem presentes dois elementos recessivos ou seja, a ausência de um dominante.

O próprio Mendel desenvolveu os conceitos de dominância e recessividade, e seu trabalho inclui uma distinção clara entre genótipo e fenótipo. O fenótipo é o que se observa (cor dos olhos, formato da semente), ao passo que só pode-se conhecer o genótipo básico por meios mais sutis. Através da genética moderna sabe-se hoje que os genes estão agrupados em uma molécula de ARN (Ácido Ribonucléico) dentro dos cromossomos, que por sua vez estão situados no núcleo das células.

Dessa forma uma parte faltante na teoria de Darwin foi preenchida pela genética, e a união de ambas ficou sendo conhecida como Neodarwinismo, que é a teoria mais aceita nos dias de hoje para explicar questões como a da biodiversidade.

Demais informações sobre o tema podem ser encontradas em (Green, 2006).

2.1.2 Inteligência de Máquina

Inteligência pode ser definida como a capacidade de um sistema de adaptar seu comportamento para encontrar metas desejadas em uma gama de ambientes (Fogel, 2006).

Inteligência artificial é uma área de pesquisa que tem por objetivo simular em sistemas computacionais a capacidade humana de resolver problemas, não implicando, necessariamente, que a máquina "pense" da mesma forma que um ser humano.

De acordo com Piennar (Piennar, 1998) o fundamental para a inteligência, seja biológica ou de máquina, é o método pelo qual a informação é organizada para criar representações internas. Essas representações são por sua vez processadas por um sistema visando satisfazer as metas condicionais.

O professor Herbert A. Simon definiu a "ciência do artificial" como a maneira pela qual um sistema de tratamento de informação pode representar a informação coletada do mundo exterior, e utilizá-la para elaborar suas próprias ações (Pessis-Pasternnak *et al.*, 1992).

A evolução tem criado criaturas de crescente inteligência o tempo todo (Bäck *et al.*, 2000).

2.1.3 Algoritmos Evolutivos

A essência dos algoritmos evolutivos está contida na informática bio-inspirada, uma linha de pesquisa que visa obter soluções para problemas computacionais complexos através do estudo de sistemas biológicos naturais.

Ao longo do tempo, graças a teoria Neodarwinista, surgiram vários algoritmos evolutivos que são aplicados às mais diversas áreas.

Desde a década de 40, cientistas da computação têm tentado usar mecanismos naturais como metáforas para a computação. Redes Neurais Artificiais certamente são o exemplo mais popular. No final dos anos 50 cientistas começaram a pesquisar os princípios da evolução biológica visando desenvolver novos modelos para a computação, dando assim os primeiros passos para o que viria a se tornar a Computação Evolutiva. Problemas de otimização têm sido o principal foco de aplicação desta nova metodologia. Evolução é um processo de otimização (Mayr, 1988).

Algoritmos Evolutivos são tipicamente aplicados para resolver problemas de busca na

forma:

$$f : S \rightarrow \mathbb{R} \tag{2.1}$$

onde S é um espaço de busca que é constituído por todas as possíveis soluções para um problema em particular (Zebulum *et al.*, 2002). Dependendo das particularidades do problema, as soluções podem ser representadas por vetores de n-dimensionais, binários, inteiros ou de números reais. Para todas as soluções existentes em S um número real é associado, chamado de valor de *fitness* (Equação 2.1), provendo um modo de medir o quão adequada a solução é para resolver tal problema. Em outras palavras o valor associado ao *fitness* mostra o quanto este indivíduo está adaptado ao meio ambiente (problema). A primeira tarefa de um algoritmo evolutivo é a de amostrar eficientemente um amplo espaço de busca S encontrando soluções em conformidade com o objetivo do problema.

Algoritmos evolutivos utilizam o processo de aprendizado coletivo de uma população de indivíduos, o que implica em um paralelismo na busca por soluções. Usualmente, cada indivíduo representa (ou codifica) um ponto de busca no espaço de potenciais soluções para um dado problema. A essa população é aplicado um processo de evolução, permitindo através de várias gerações, o surgimento de soluções melhores que as anteriores. Algoritmos Evolutivos não são garantia para a obtenção de soluções ótimas, ao lidar-se com amplos e complexos espaços de busca, a otimalidade pode ser algo de difícil obtenção sendo possível esperar-se soluções satisfatórias.

Aplicações envolvendo Computação Evolutiva recaem em uma ampla gama de áreas, tais como planejamento, projeto, simulação e identificação, controle e classificação (Bäck *et al.*, 2000). Os principais algoritmos evolutivos disponíveis atualmente são:

- Algoritmos Genéticos (AG)
- Estratégias Evolutivas (EE)
- Programação Evolutiva (PE)
- Programação Genética (PG)
- Sistemas Classificadores (SC)

Com base na teoria neo-Darwiniana da evolução das espécies, é possível propor um algoritmo evolutivo básico ou padrão com as seguintes características:

- Uma população de candidatos a darem solução (denominados indivíduos ou cromossomos) que se reproduz com herança genética. Onde cada um dos indivíduos da população corresponde a uma estrutura de dados que representa ou codifica um ponto em um espaço de busca. Os indivíduos se reproduzem de forma sexuada ou assexuada, gerando filhos que terão parte de seu material genético proveniente de seu(s) pai(s). Na reprodução sexuada em particular existe a troca de material genético (crossover ou recombinação) entre dois ou mais indivíduos pais. Já no caso da reprodução assexuada o filho ou os filhos serão cópias com variações que os diferenciarão de seus genitores.

- Variação genética: durante o processo reprodutivo os filhos, não apenas herdam as características paternas, como também podem sofrer mutações responsáveis por alterações em seu código genético.

- Seleção Natural: a avaliação dos indivíduos em seu ambiente através de uma função de avaliação ou *fitness* deste indivíduo. Os valores individuais de *fitness* são a base que resultará em uma competição pela sobrevivência e reprodução no ambiente. Os indivíduos com altos valores de *fitness* têm maior vantagem frente a indivíduos com *fitness* menores.

Devido ao foco do trabalho, os AG's serão abordados com maior ênfase. Demais informações a respeito dos algoritmos evolutivos citados até então podem ser encontradas em (Bäck *et al.*, 2000) e (Zebulum *et al.*, 2002).

Algoritmos Genéticos

Algoritmos genéticos são uma classe de algoritmos estudada e analisada primeiramente por John Holland (Holland, 1975). Seus objetivos eram a investigação de processos adaptativos em sistemas naturais, e a criação de programas de computador que mostrassem comportamento similar com respeito aos sistemas naturais investigados. Assim como os demais, estes algoritmos também possuem inspiração biológica utilizando-se de conceitos extraídos da teoria neodarwinista, como a genética.

AGs tornaram-se os mais populares algoritmos evolutivos devido à bem-sucedida aplicação na otimização de problemas de busca, particularmente naqueles problemas em que o tamanho ou a complexidade do espaço de busca torna impraticável o uso de outras técnicas de otimização (Zebulum *et al.*, 2002).

Existem três características que distinguem um AG dos demais algoritmos evolutivos: representação dos indivíduos utilizando *bitstrings*; método de seleção dos indivíduos proporcio-

nal ao seu valor de *fitness*; e o método primário de produção de variação e cruzamento genético (*crossover*) (Bäck *et al.*, 2000). Dentre as três, contudo, a ênfase situada no *crossover* é que faz a distinção do AG frente aos demais algoritmos evolutivos. Algumas subseqüentes implementações de algoritmos genéticos tem optado por utilizar métodos alternativos de seleção tais como seleção por torneio ou rank e representações diferentes de *bitstrings*, como por exemplo a representação de indivíduos com valores inteiros ao invés de valores binários, em virtude de um melhor tratamento de problemas. Ao agrupamento de genes que representa uma solução no espaço de busca dá-se o nome de cromossomo. Exemplos de cromossomos podem ser vistos na figura 2.3 .

Figura 2.3: Cromossomos

Onde locus é a posição do gene no cromossomo e os alelos os valores que o gene pode assumir.

Mecanismos de Seleção, Cruzamento e Mutação

Os operadores genéticos utilizados no AG clássico como mecanismos evolutivos são o *crossover* de ponto único, a mutação e a seleção probabilística por roleta.

A seleção por roleta é um método que atribui a probabilidade de escolha de um indivíduo (para cruzamento e conseqüente propagação de material genético) ao seu valor de *fitness*, em outras palavras indivíduos com *fitness* mais elevados terão uma fatia maior da roleta, como mostrado na figura 2.4 . A roleta é um gerador de números pseudo-aleatórios com distribuição uniforme, com números gerados dentro do intervalo dado pela soma dos valores de *fitness* de todos os indivíduos.

O operador de *crossover* realiza a mistura do material genético de dois pais pré-selecionados pelo operador de seleção dando origem a dois novos filhos, criando uma nova geração com novos indivíduos que possuem herança genética proveniente dos indivíduos pais (figura 2.5). Após esses indivíduos sofrerão pequenas modificações através de um processo de mutação.

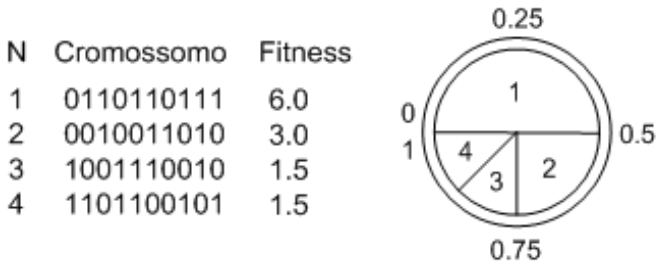

Figura 2.4: Método da Seleção por Roleta

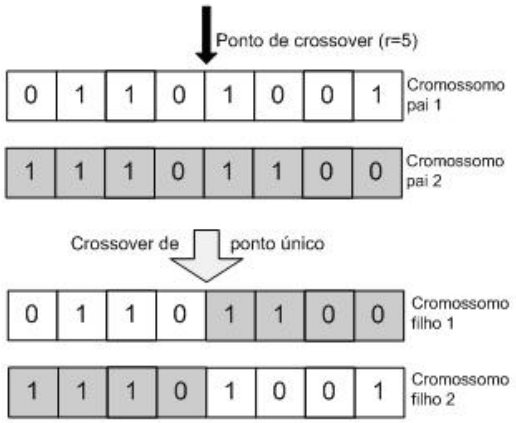

Figura 2.5: Crossover de ponto único

O operador de mutação realiza pequenas mudanças nos indivíduos da nova geração com o intuito de promover maior diversidade (figura 2.6) e não estagnar o processo evolutivo, o que vem a ser muito interessante para uma melhor exploração do espaço de busca evitando que a busca estagne em ótimos locais.

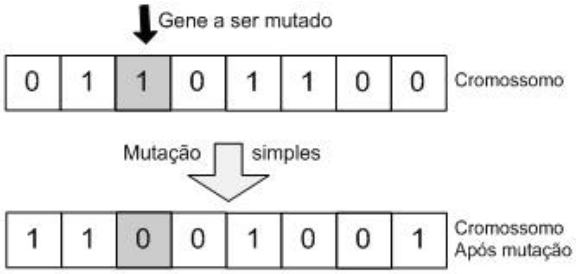

Figura 2.6: Mutação simples

As etapas de maior importância na utilização de algoritmos evolutivos, e conseqüentemente as que requerem maior atenção, são a codificação (a forma como as soluções serão representadas no AG) e a função de *fitness* que avaliará as soluções codificadas. Codificações e

funções de *fitness* mal elaboradas podem comprometer todo o processo evolutivo dentro de um AG, ou qualquer outro algoritmo evolutivo.

O diagrama da figura 2.7 mostra o fluxo básico dos algoritmos genéticos.

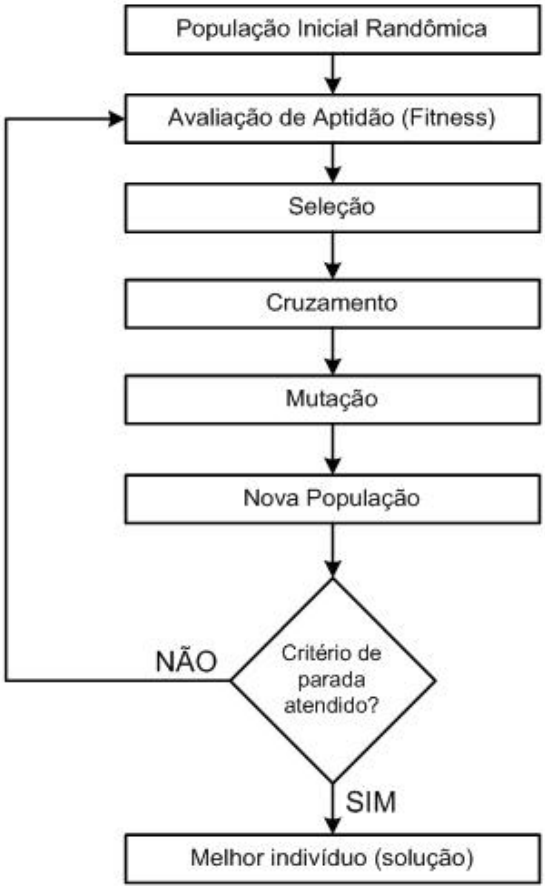

Figura 2.7: Diagrama de fluxo dos AG's

Os algoritmos genéticos são projetados para a simulação de evolução, não propriamente para resolver problemas. Contudo, a evolução traz consigo alguns projetos interessantes, mas deve-se ter em vista que, de fato, a evolução é um processo oportunista operando em um ambiente que está em constante mudança (Bäck *et al.*, 2000).

2.2 *Hardware* Evolutivo

2.2.1 Síntese de *Hardware* Digital

Projetos de sistemas digitais podem ser definidos como a transformação de uma descrição inicial do sistema (especificação) em uma descrição final. Isso se dá através de sucessivas etapas podendo envolver diversos tipos de níveis de abstração, onde a principal diferença está no fato desta última conter todas as informações necessárias a sua fabricação (Calazans, 1998). A manipulação destas etapas, chamada de projeto de *hardware*, pode ser classificada como síntese, extração, validação e otimização (Suzim, 1998), e pode ser observada no diagrama da figura 2.8.

Figura 2.8: Modelo de representação do processo de projeto de sistemas digitais proposto por Suzim

A operação de síntese é a tradução de uma descrição em um dado nível para uma descrição em nível inferior, que ocorre através do acréscimo de informação permitindo a criação de uma descrição menos abstrata. Já a operação de extração, gera uma descrição mais abstrata, com menos detalhes e características de implementação. Tanto a operação de síntese quanto a operação de extração ocorrem sobre descrições de um mesmo nível.

Pode-se encontrar diversas metodologias para representar um projeto de *hardware* sendo a mais comum o Diagrama Y, proposto inicialmente por Gajski e Kuhn (Gajski e Kuhn, 1983). Este diagrama tem o objetivo de estratificar o processo de projeto de sistema digitais em diferentes níveis de abstração e domínios de descrição (figura 2.9).

Nesta representação, os círculos correspondem aos níveis de abstração e os segmentos de retas correspondem aos domínios de descrição.

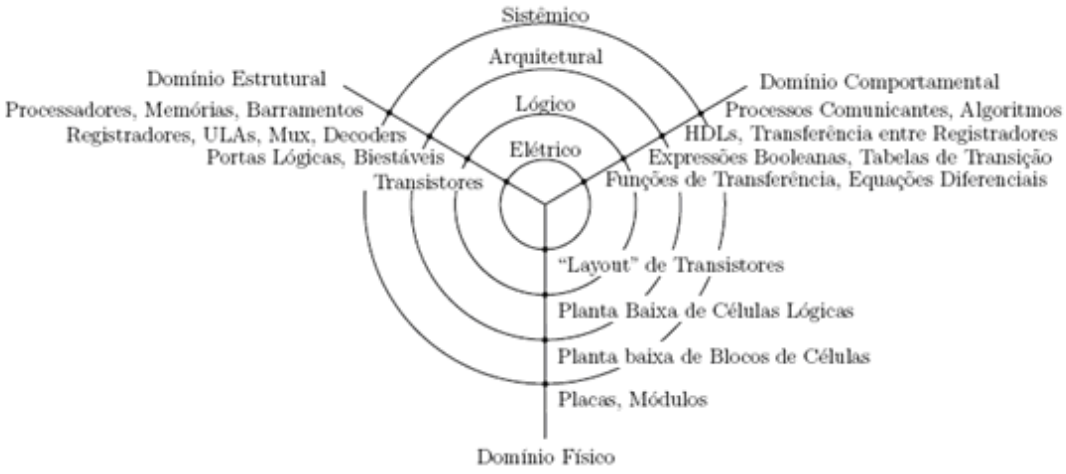

Figura 2.9: Diagrama Y

Sistemas digitais podem ser geralmente classificados em quatro níveis diferentes de abstração: nível elétrico, como por exemplo, transistores e resistores, nível lógico aonde se encontram as portas lógicas, *flip-flops* e equações booleanas, o nível arquitetural aonde se pode exemplificar com ULAs e multiplexadores, e os níveis sistêmicos representados processadores, memórias e afins.

Além dos níveis de abstração é possível classificar as descrições em relação ao tipo de informação contida nelas. Dessa forma as descrições podem ser classificadas em três tipos diferentes: Físico, Comportamental e Estrutural. O tipo físico contém informações sobre os componentes/módulos, o tipo comportamental possui informações referentes ao comportamento do sistema, e o tipo estrutural que contém informações de interconexão entre os diversos blocos.

A região central do Diagrama Y corresponde à descrição onde se encontram todas as informações necessárias à fabricação do sistema, também chamada de descrição final. Cada intersecção de um círculo com um segmento de reta representa um tipo de descrição diferente. Automatizar as etapas de descrições acima é algo imprescindível na redução do tempo de projeto. Entretanto, à medida em que a escala de integração aumenta cresce a dificuldade em se obter projetos livres de erros (Calazans, 1998). Dessa forma, ferramentas para automatizar o processo de projeto deverão sofrer constante evolução.

2.2.2 Justificativas Vantagens e Aplicabilidade

Com a evolução da eletrônica, circuitos integrados cada vez mais complexos e de grande escala de integração são produzidos visando atender um mercado interessado em dispositivos onde muitas vezes tamanho reduzido e baixo consumo são premissas inevitáveis.

A automação vem sendo usada na síntese de circuitos por muitos anos. Um projeto simples e tradicional de circuitos digitais envolve o desenvolvimento de um circuito aplicado a uma tecnologia específica com a utilização de técnicas de minimização, além de algumas regras de posicionamento e de roteamento (*placement and routing rules*) (Campos *et al.*, 2006).

Em decorrência do crescimento da indústria eletrônica e de problemas de projeto tais como escalabilidade, roteamento, tempo de desenvolvimento, confiabilidade entre outros, técnicas utilizadas para o projeto de circuitos no passado tornaram-se obsoletas, sendo que as técnicas clássicas atuais muitas vezes não conseguem manter uma taxa de crescimento análoga. Na atualidade projetos de *hardware* exigem a capacidade de sintetizar circuitos cada vez mais complexos, gerando a necessidade de técnicas mais aprimoradas, que resolvam problemas combinatórios com alto grau de complexidade e em tempo hábil.

Dessa forma, o *hardware* evolutivo ou EHW (*Evolvable Hardware*) propõe a automatização de circuitos eletrônicos visando prover soluções mais generalizadas. EHW é uma área de pesquisa recente, formalizada em 1997 em um encontro de pesquisadores na Universidade de Napier, Escócia (Zebulum *et al.*, 2002). Ela propõe a utilização de algoritmos evolutivos para o projeto de circuitos eletrônicos, unindo eletrônica e Computação Evolutiva com o objetivo de automatizar o processo e síntese de circuitos digitais ou analógicos, a partir de sua descrição comportamental (figura 2.10).

Figura 2.10: Processo EHW

Usar evolução para o projeto de circuitos pode trazer benefícios importantes como à automação e a descoberta de novos circuitos e arquiteturas não antes realizadas para uma escala crescente de aplicações. Além dessas técnicas poderem minimizar consumo, volume e custo dos circuitos projetados.

Eletrônica Evolutiva lida com enormes espaços de busca, requerendo, portanto, sofisticadas técnicas para tal. Naturalmente, quando o espaço de busca é muito amplo, somente buscas aleatórias têm alguma chance de ter sucesso (Zebulum *et al.*, 2002). Contudo, alguns procedimentos devem ser seguidos:

- O espaço de busca amostrado pelo algoritmo deve ter tamanho limitado. Embora seja importante amostrar uma ampla variedade de topologias, alguns critérios devem ser escolhidos para controlar o número de possíveis soluções.
- Geralmente é necessário adaptar as técnicas de busca para as particularidades do projeto problema.

Pode-se descrever as mais notáveis características de circuitos evoluídos observada em pequena ou grande escala como sendo:

- Potencial de encontrar novos circuitos.
- Possibilidade de encontrar novas regras de projeto a partir de novos circuitos obtidos.
- Métodos evolutivos podem contemplar um grande conjunto de especificações de projeto comparado com técnicas de projetos feitos por humanos.
- Sistemas evolutivos são hábeis em conseguir circuitos competitivos quando comparado a outros circuitos existentes no estado atual da arte da eletrônica.

Dentre as características citadas acima, a terceira deve ser enfatizada. Atualmente, é importante conseguir circuitos eletrônicos que atendam a muitas especificações: bom desempenho; área reduzida; baixo consumo de energia; velocidade; e em alguns casos, tolerância a falhas. Métodos convencionais de projeto geralmente não são apropriados quando muitas especificações são consideradas.

Algoritmos evolutivos podem explorar algumas regiões do espaço de projeto que estão além do escopo dos métodos convencionais, como por exemplo o mapa de Karnaugh ((Karnaugh, 1953)) e a álgebra Booleana.

2.2.3 Funcionamento da Técnica

Em *Hardware* Evolutivo um algoritmo evolutivo, geralmente o algoritmo genético, é aplicado a uma população de indivíduos (soluções candidatas) codificados na forma de cromossomos, onde cada um deles representa um circuito eletrônico. Usualmente esses indivíduos são *strings* binárias de comprimento fixo e estão contidos em uma população de tamanho constante. É importante salientar que outras abordagens utilizando outros algoritmos evolutivos podem ser encontradas, como a utilização de programação evolutiva por Coello (Coello *et al.*, 1996) onde os autores utilizam uma codificação em árvore baseada em multiplexadores para síntese de funções lógicas. Outra abordagem é a proposta por Levi (Levi, 2000), utilizando um algoritmo híbrido intitulado HereBoy.

Ao se iniciar o processo evolutivo, os valores internos dos cromossomos da primeira população são gerados aleatoriamente. A avaliação dos cromossomos (*fitness*) é obtida através da evolução extrínseca, sendo os circuitos simulados em computador, ou através de evolução intrínseca, aonde os circuitos são implementados em dispositivos físicos reconfiguráveis (usualmente FPGA's), e mais recentemente em evolução mixtrínseca onde os circuitos são evoluídos parte em software e parte em *hardware*. A aptidão do circuito em atingir seu objetivo é avaliada através da inserção de um conjunto de determinados valores de entrada, e a observação das respostas dos sinais de suas saídas. O operador de seleção presente no processo é probabilístico, de forma que os cromossomos com maior aptidão terão maiores chances de serem selecionados. Após isso, os operadores evolutivos são aplicados aos cromossomos selecionados, tais como *crossover* e mutação para a geração da nova população.

De forma geral, EHW é um processo exploratório de busca em um domínio pré-estabelecido, visando obter o indivíduo com as melhores características dentre todas as iterações efetuadas pelo algoritmo.

A utilização de algoritmos evolutivos no projeto de circuitos eletrônicos, como visto neste capítulo, é uma ferramenta alternativa que pode ser muito interessante em algumas aplicações de engenharia. O próximo capítulo aborda o tema de circuitos lógicos reconfiguráveis, linguagem descritiva de hardware e noções de controle com aplicação ao problema do pêndulo.

Capítulo 3

LÓGICA RECONFIGURÁVEL E A LINGUAGEM DE DESCRIÇÃO DE *HARDWARE* VHDL

3.1 Field Programable Gate Array

Os *chips* FPGA (Field Programable Gate Array) são circuitos eletrônicos integrados que possuem uma engenharia de reconfiguração interna permitindo a implementação de circuitos lógicos. Esses chips pertencem à classe dos dispositivos lógicos reprogramáveis.

O termo dispositivo lógico programável (*Program Logic Device* - PLD) refere-se a qualquer tipo de circuito usado para implementar circuitos digitais, onde o dispositivo pode ser configurado pelo usuário para realizar uma vasta gama de projetos (Brow e Rose, 1996) . A configuração de um PLD freqüentemente envolve a colocação do componente em uma unidade especial de programação, mas alguns componentes também podem ser configurados pelo próprio sistema (*in-system*).

Os PLD's para propósitos gerais com maior densidade disponíveis na atualidade são os conhecidos como FPGA's. Os FPGA's são constituídos por um conjunto de elementos de circuitos, chamados de elementos lógicos (logic elements - LE's), circundados por blocos de entrada e saída (E/S) (figura 3.1). Os LEs implementam funções lógicas simples, e seus blocos de E/S são conectados entre si por meio de interconexões programáveis. Essas interligações programáveis permitem que diversos LE's possam ser conectados para implementar circuitos

sequenciais e/ou combinacionais complexos.

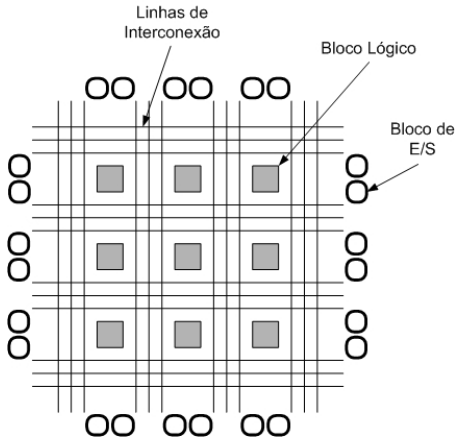

Figura 3.1: Estrutura Básica de um FPGA

3.1.1 Projetos Utilizando Dispositivos Lógicos Programáveis

O projeto de sistemas digitais destinados a implementação em dispositivos lógicos programáveis é realizado por ferramentas que pertencem a uma classe denominada de EDA (*Eletronic Design Automation*). Uma ferramenta de EDA para PLD típica inclui programas para as seguintes tarefas: entrada inicial para o projeto, otimização, adequação ao dispositivo, simulação e configuração (figura 3.2).

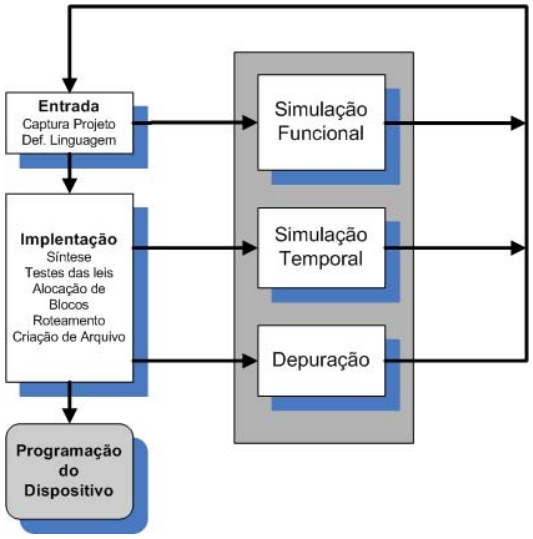

Figura 3.2: Fluxo de projeto lógico programável

Existem algumas ferramentas computacionais usuais e apropriadas para a realização da entrada de projeto. Alguns projetistas preferem a entrada através de esquemáticos, enquanto outros preferem o uso das linguagens de descrição de hardware (*Hardware Description Languages* - HDL), tais como Verilog e VHDL.

Tradicionalmente, as ferramentas baseadas em diagramas esquemáticos propiciam maior controle sobre o particionamento e posicionamento da lógica do dispositivo. Porém, tal benefício vem em detrimento do tempo gasto na realização do projeto, uma vez que ele é executado no nível de portas lógicas (Nicolato, 2002). Em contrapartida, as ferramentas baseadas em linguagens HDL permitem que o projeto seja realizado num nível de abstração mais elevado (por ex. descrição comportamental em VHDL), possibilitando maior agilidade no processo de desenvolvimento de projetos digitais. As desvantagens comuns a tais linguagens são a diminuição do desempenho e a densidade lógica necessária (quantidade de lógica implementada por área do componente).

Atualmente, os principais fabricantes de PLDs disponibilizam bibliotecas de funções (em alguns casos, parametrizadas) básicas tais como somadores, multiplicadores, multiplexadores, memórias etc, que podem ser "chamadas" a partir de editores de esquemáticos ou instanciadas por meio de HDL. Adicionalmente, há uma ampla pesquisa sobre tradutores que fazem a conversão de programas escritos em linguagem de alto nível para níveis lógicos inferiores (Sankaran e Haggard, 2001).

A verificação ocorre em vários níveis, e durante vários passos do procedimento, dependendo dos critérios adotados pelo projetista. Existem alguns tipos básicos de verificação quando são utilizados dispositivos de lógicas programáveis. A simulação funcional é realizada em conjunto com a entrada do projeto, mas antes do posicionamento e roteamento, objetivando a verificação de sua funcionalidade lógica. A simulação considerando as características de temporização é realizada em uma etapa posterior ao posicionamento e roteamento. Nesse passo, o programa de desenvolvimento determina os atrasos do circuito, possibilitando a verificação completa de sua temporização.

Uma boa técnica para projetos com lógica programável é, primeiramente, realizar uma simulação funcional para determinar o funcionamento correto do circuito, verificar a temporização e, finalmente, verificar a funcionalidade completa testando-o no sistema, incluindo dispositivos físicos e as exigências ambientais da aplicação (Nicolato, 2002).

Muitos dispositivos de lógica programável têm a grande vantagem de poderem ser programados, ou re-programados, no sistema, isto é, não necessitam de uma unidade especial de

programação. Assim, o projeto pode ser facilmente verificado no sistema real, reduzindo a necessidade da criação de vetores de simulação muito complexos.

Depois da criação do arquivo de programação, o dispositivo é configurado e estará pronto para operar. O método de programação depende da tecnologia dos componentes alvo. Algumas tecnologias, por exemplo as PROMs (*Programable Read Only Memory*), requerem algum tipo de dispositivo de programação. Dispositivos que podem ser programados no sistema nem sempre necessitam de um programador físico externo, mas requerem algum tipo de recurso para carregar o arquivo de programação no *chip*. Isso pode ser realizado com o auxílio de um microprocessador, microcontrolador ou via portas no padrão JTAG.

Seja qual for o fabricante ou a tecnologia, é importante ressaltar que o conhecimento da arquitetura do componente, bem como das ferramentas de projeto são muito importantes para obtenção de bons resultados.

3.2 A Linguagem VHDL

A linguagem VHDL (Very High Description Language) é uma linguagem usada para descrever a arquitetura e o comportamento de circuitos eletrônicos. Essa linguagem representa uma alternativa à descrição de circuitos através de diagramas elétricos ou esquemáticos.

Historicamente, o desenvolvimento da linguagem de descrição de circuitos VHDL deveu-se à necessidade de uma ferramenta de projeto e documentação padrão para o projeto VHSIC (Very High Speed Integrated Circuit), do Departamento de Defesa dos Estados Unidos da América (DARPA). No começo da década de oitenta, mas precisamente em 1981, esse departamento patrocinou um encontro de especialistas para discutir métodos para descrição de circuitos. Em 1983, o Departamento de Defesa definiu os requisitos de uma linguagem de descrição de circuitos padrão e concedeu às firmas “IBM”, “Texas” e “Intermetrics” um contrato para o desenvolvimento da linguagem e ferramentas.

VHDL suporta projetos com múltiplos níveis de hierarquia, dessa maneira a descrição pode consistir na interligação de outras descrições menores, a um código que representa o comportamento esperado do circuito. Esses estilos são denominados estrutural e comportamental, e podem ser mesclados em uma mesma descrição. A estrutura hierárquica e a opção de combinar diversos estilos de descrição facilitam a condução de projetos complexos que partem de um nível mais elevado para um nível mais baixo de especificação, conhecidos como top down design ((d’Amore, 2005)).

3.2.1 Entidade de Projeto

Uma entidade de projeto, ou design entity, pode representar desde uma simples porta lógica a um sistema completo, e é composta de duas partes: declaração da entidade (Entity) e arquitetura (architecture) . A declaração de entidade especifica a interface entre a entidade e o ambiente exterior, como, por exemplo, entradas e saídas. A arquitetura possui a definição das relações entre as entradas e saídas de uma entidade. A figura 3.3 ilustra uma representação de circuito lógico em VHDL exibindo um código básico e as relações entidade/arquitetura com o circuito descrito por ele. A analogia normalmente feita com um projeto em captura esquemática, objetiva comparar a declaração da entidade ao símbolo de um bloco, e a arquitetura ao esquema contido no bloco.

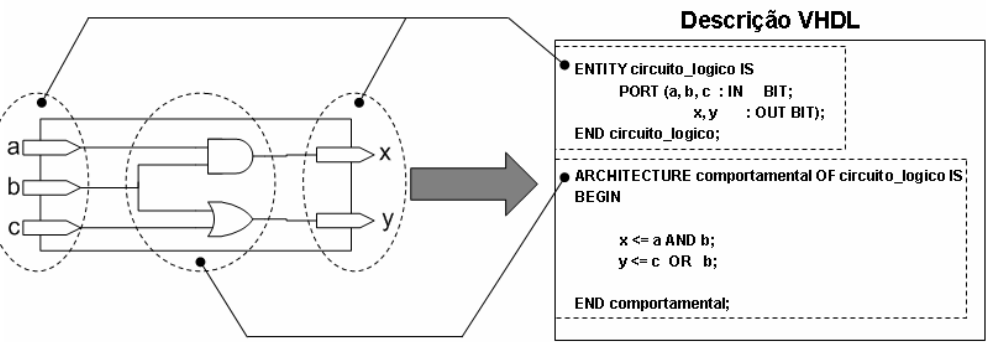

Figura 3.3: Descrição VHDL de Circuito Lógico

A linguagem VHDL é estruturada e tem sua sintaxe semelhante à linguagem de programação Pascal. Com exceção de regiões específicas, todos os comandos são executados de forma concorrente, ou seja, paralelamente. Isto significa que a ordem em relação a apresentação dos comandos é irrelevante para o comportamento da descrição. A ocorrência de um evento em um sinal leva à execução de todos os comandos sensíveis àquele sinal, da mesma forma que, em um circuito, a mudança de um valor em um determinado nó afeta todas as entradas ligadas a esse ponto do circuito.

Como o código é concorrente, a mudança de valor em "b" na figura 3.3, leva à execução em paralelo dos comandos contidos nas linhas 8 e 9. A ordem da avaliação dos comandos executada pelo simulador é irrelevante, e o resultado será sempre o mesmo. As operações internas executadas pelo simulador, contudo, são efetuadas de forma seqüencial. Dessa forma, a ferramenta de simulação necessita de um mecanismo interno para armazenar o resultado de cada comando, até que a avaliação de todos os comandos envolvidos tenho sido realizada.

A linguagem VHDL também permite delimitar regiões de código executados sequencialmente, onde a execução dos comandos se dá na ordem de sua apresentação no código. Para essas regiões são usados comandos específicos, que não podem ser empregados na região de código concorrente. Subprogramas e processos são regiões de código seqüencial.

3.2.2 Síntese de Circuitos Usando VHDL

Originalmente, VHDL não foi criada para a síntese de circuitos digitais; assim, nem todas as construções definidas são suportadas pelas ferramentas de síntese. As limitações devem-se a falta de correspondência da construção com um circuito digital, à impossibilidade da síntese com precisão, ou à falta de detalhamento para um síntese direta. Essas limitações não devem ser consideradas como um problema da ferramenta de síntese ou da linguagem, mas sim uma falha na própria descrição que está muito afastada de um circuito real ((d'Amore, 2005)). Por exemplo, um projetista pode descrever no código um elemento de memória sensível tanto a transição de subida quanto a transição de descida de um sinal de clock. Do ponto de vista do simulador VHDL, a descrição pode ser simulada sem problemas; entretanto, a ferramenta de síntese poderá não ter sucesso em sintetizar a descrição devido a ausência de um elemento desse tipo no mundo real. Os três passos em um projeto empregando uma linguagem de descrição de hardware podem ser vistos na figura 3.4.

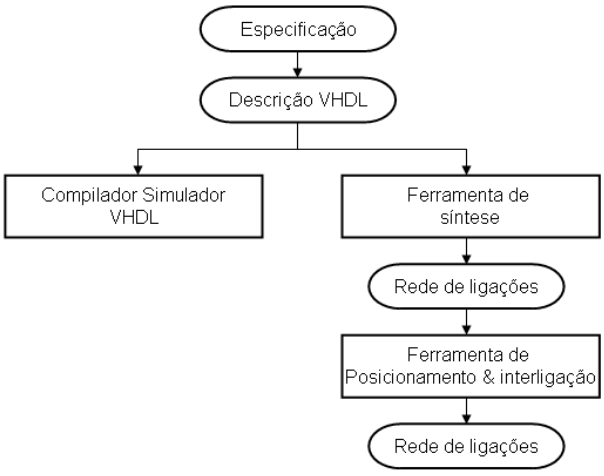

Figura 3.4: Etapas de um projeto

A partir da especificação de um projeto, é gerada uma descrição VHDL, que é submetida a um simulador para a verificação da correspondência entre a especificação e o código. A mesma descrição é interpretada por uma ferramenta de síntese que infere as estruturas necessárias para

um circuito que corresponda à descrição. Como resultado dessa etapa obtém-se um arquivo contendo uma rede de ligações de elementos básicos disponíveis na tecnologia do dispositivo empregado. Esses elementos atuam como blocos construtivos realizando a implementação física do modelo descrito pela linguagem. O arquivo contendo a rede de ligações é a base de dados para a ferramenta que realiza o posicionamento e a interligação dos componentes, place and route. A saída da ferramenta de posicionamento e interligação é um arquivo contendo os dados necessários para confecção do circuito no dispositivo usado para a síntese.

Para um maior aprofundamento sobre o tema de síntese de circuitos por VHDL são sugeridas as bibliografias de d'Amore (2005) e Perry (2002).

Neste capítulo foram abordados os temas de hardware reconfigurável (FPGAs) e linguagem de descrição de hardware (VHDL), ambas ferramentas amplamente usadas na implementação do sistema de controle que será detalhado nos próximos capítulos.

Capítulo 4

CONTROLE APLICADO AO PÊNDULO

4.1 Sistemas Automáticos

Uma definição de sistema pode ser dada como sendo todo conjunto de elementos inter-relacionados, aonde o comportamento individual de cada elemento afeta o sistema como um todo. Observe que se faz referência a elementos, não a objetos ou peças, isso porque um sistema não tem que ser obrigatoriamente físico, tangível. Sistemas podem muito bem estarem conformados por unidades de informação, leis abstratas e vários outros "elementos"não físicos.

Todo sistema físico denomina-se genericamente de planta. Toda planta tem um determinado comportamento, isto é, executa uma determinada ação (Pazos, 2002). Genericamente, representa-se a planta, sua resposta (saída y(t)) e excitação (entrada u(t)) como no diagrama da figura 4.1.

Figura 4.1: Conceito de planta

Um exemplo de sistema físico pode ser o de uma simples alavanca, aonde existe uma relação entre o torque de saída e o torque de entrada, que depende do ponto de apoio na alavanca.

4.2 Sistema de Controle em Malha Fechada

Os sistemas de controle em malha fechada, ou realimentados, são sistemas que utilizam como base o sinal de erro atuante, que é a diferença entre o sinal de realimentação (que pode ser o próprio sinal de saída ou uma função do sinal de saída e suas derivadas e/ou integrais) e o sinal de referência. O controlador objetiva minimizar o erro e ajustar a saída ao valor desejado informado no sinal de referência. O diagrama da figura 4.2 ilustra um sistema em malha fechada.

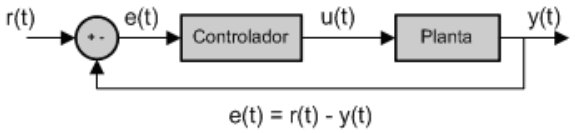

Figura 4.2: Diagrama de sistema de controle em malha fechada

Onde r(t) corresponde ao sinal de referência, u(t) o sinal de controle para a planta (sistema a ser controlado), y(t) a resposta do sistema com base no sinal de controle e e(t) o sinal de erro.

4.3 Controle do Pêndulo

Na teoria de controle existem basicamente dois tipos de variáveis em um sistema, a variável controlada e a variável manipulada. A variável controlada é a grandeza ou a condição que é medida e controlada. A variável manipulada é a grandeza ou a condição modificada pelo controlador, de modo que afete o valor da variável controlada (Ogata, 2006). A figura 4.3 ilustra esse conceito mostrando o acionamento de um motor CC por PWM.

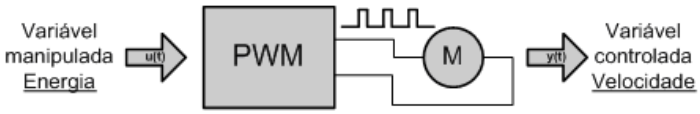

Figura 4.3: Acionamento por PWM

No sistema implementado neste trabalho o controle do pêndulo se dá através de alterações no fluxo de energia entregue a ele, que neste caso é a variável manipulada no sistema. O controle de energia resulta em ajustes na posição da haste do pêndulo, com base no comportamento do sistema dinâmico existente entra as variáveis controlada e manipulada. Dessa

maneira, a entrada do sistema (sinal de referência) é uma informação que indica a posição que a haste deve se manter, e o sistema de controle responde a essa entrada deslocando a haste até a posição especificada.

Uma abordagem mais aprofundada sobre a área de controle de sistemas dinâmicos pode ser vista em Ogata (2006) e Nise (2004).

4.4 Especificações Técnicas

Entende-se por especificações técnicas o conjunto de requerimentos ou exigências definidas pelo usuário do sistema com respeito ao seu comportamento, ou como ele quer que se comporte o sistema a controlar (Pazos, 2002). Em geral, uma primeira especificação técnica mínima exigida é que o sistema seja estável. Se o sistema não o for naturalmente, o controle deverá fazer com que se comporte como tal. Uma outra especificação pode ser o percentual de *overshoot*, ou relação entre o valor máximo da resposta sobre o seu valor final, e o seu valor final (valor quando $t \to \infty$).

4.5 Pêndulo Amortecido

Em mecânica, um pêndulo simples é um instrumento ou uma montagem que consiste em um objeto que oscila em torno de um ponto fixo, realizando um movimento bidimensional no plano xy caracterizado por uma função seno. O braço ou haste executa movimentos alternados em torno da posição central, chamada posição de equilíbrio (figura 4.4).

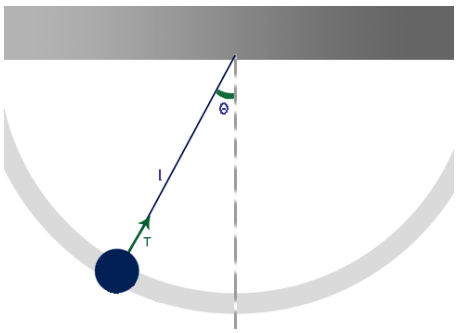

Figura 4.4: Ilustração de pêndulo simples

O termo “amortecido” é dado ao sistema de movimento oscilatório cujo movimento sofre

resistência devido a uma força contrária de atrito, como é o caso do pêndulo como objeto mecânico real.

O pêndulo é um modelo clássico utilizado no estudo de juntas robóticas de sistema manipuladores e como excelente meio para testes de técnicas de controle avançadas para sistemas não lineares. O modelo matemático de movimento do pêndulo pode ser expresso pela equação 4.1.

$$ml^2\theta'' + kl^2\theta' + mgl\sin(\theta) = \tau \tag{4.1}$$

Sendo m a massa concentrada em seu centro de massa, g a aceleração da gravidade, k a viscosidade do meio e l o comprimento da haste do pêndulo.

4.6 Controlador Proporcional, Integral e Derivativo

Este tipo de controle consiste em uma estratégia onde é aplicada à planta um sinal de excitação proporcional ao erro, somado à sua função derivada e sua função integral.

O controle PID é a estratégia de controle mais genérica e provavelmente uma das mais utilizadas. Fornece resposta rápida, bom controle de estabilidade do sistema e baixo erro de regime permanente para sistemas de até segunda ordem. Tais vantagens acontecem porque o controle PID permite adaptar o sistema realimentado geral (sistema planta-controlador), quase que idealmente seja qual for o modelo da planta (Pazos, 2002).

O PID discreto pode ser implementado de diferentes formas. As estruturas PIDs discretas mais comuns são o controlador proporcional (equação 4.2), controlador Proporcional-Derivativo (equação 4.3), controlador proporcional Integral (equação 4.4) e controlador proporcional - integral - derivativo (equação 4.6).

$$u(n) = Kpe(n) \tag{4.2}$$

$$u(n) = \left[Kp + \frac{Kd}{Ts}\right] e(n) - \frac{Kd}{Ts} e(n-1) \tag{4.3}$$

$$u(n) = u(n-1) + \left[Kp + \frac{KiTs}{2}\right] e(n) + \left[-Kp + \frac{KiTs}{2}\right] e(n-1) \tag{4.4}$$

$$u(n) = u(n-1) + \left[Kp + \frac{Kd}{Ts} + \frac{KiTs}{2}\right] e(n) + \left[\frac{KiTs}{2} - Kp - 2\frac{Kd}{Ts}\right] e(n-1) + \frac{Kd}{Ts} e(n-2) \tag{4.5}$$

sendo Kp,Kd e Ki os ganhos proporcional, diferencial e integral, respectivamente. Ts é o tempo de amostragem e $e(n)$ descreve o sinal de erro, dado por:

$$e(n) = r(t) - y(t) \tag{4.6}$$

onde r(t) é o sinal de referência e y(t) é a variável do sistema.

O desempenho do controlador PID é dependente da sintonia de seus parâmetros, que em alguns casos pode ser uma tarefa complicada devido à presença de comportamento não linear na planta.

4.7 Técnica Utilizada no Projeto dos Controladores

Os controladores implementados aqui (Kp e EHW) são fruto do trabalho de doutorado desenvolvido por Campos (2007) cujo um dos objetivos foi o de aplicar técnicas evolutivas no projeto de controladores para pêndulo amortecido.

A função de fitness, responsável por avaliar as soluções com base em seu comportamento, foi definida utilizando um índice de desempenho baseado na integral do erro quadrático ISE (*Integral of the Square Error*) Eq. 4.7 usada como critério de minimização dos controladores desenvolvidos naquele trabalho. Esse índice foi escolhido por penalizar fortemente valores altos na variável de erro, e por possuir um bom tratamento matemático (Campos, 2007).

$$fitness = f(J_{ISE}) = \frac{1}{1 + J_{ISE}} \tag{4.7}$$

O controlador proporcional Kp teve seu ganho sintonizado através de um algoritmo genético utilizando uma codificação binária tradicional (figura 4.5). No projeto do controlador evolutivo, apenas o critério funcional foi usado como elemento de projeto, ou seja, apenas as entradas e as saídas produzidas pelo sistema foram consideradas, numa abordagem chamada de caixa preta. A estrutura interna do circuito digital do controlador não foi considerada, desse modo apenas ações que produzem uma saída notável são cobertas pela função de fitness do AG

(Campos, 2007).

$$(K_P, K_D, K_I) = \underbrace{010...00}_{K_P}\underbrace{101...01}_{K_D}\underbrace{111...10}_{K_I}$$

Figura 4.5: Codificação dos ganhos do controlador

A codificação adotada para o controlador EHW consiste em um cromossomo binário formado por um conjunto de n segmentos, cada segmento representa uma porta lógica e é um nó do circuito digital formado pela função booleana, além de conter as informações sobre os nós que devem ser conectados às entradas da porta lógica por ele representada. Cada segmento é formado por uma palavra de 15 bits, sendo que os 3 primeiros bits designam o tipo de porta ou função do nó, seguido de um segmento de 6 bits que representa a conexão da primeira entrada e outro segmento de mesmo tamanho representando a conexão da segunda entrada. A biblioteca de portas suportada nos cromossomos é composta pela portas *BUFFER*, INV, AND, NAND, OR, NOR, XOR e XNOR.

A topologia é especificada por uma lista de tipos de componentes junto com seus nós terminais (PSPICE netlist). Os componentes não conectados são desconsiderados, o que possibilita ter um número de componentes variáveis em um circuito, embora o tamanho do cromossomo seja fixo. A figura 4.6 exibe uma ilustração do método de codificação para um circuito simples de 4 entradas.

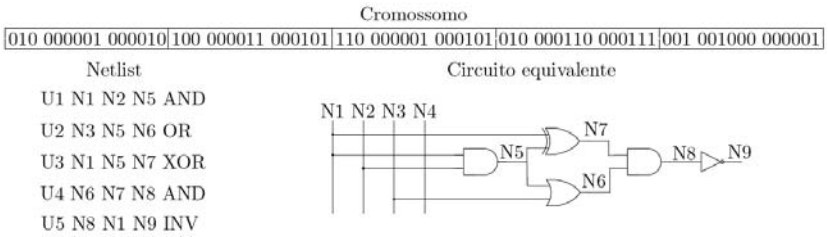

Figura 4.6: Codificação netlist para controlador EHW

O primeiro passo na criação do circuito evolutivo é a geração de uma população inicial aleatório de cromossomos. Na evolução extrínseca os cromossomos são convertidos em um modelo simulado (SPICE) e são avaliados de acordo com as respostas geradas, por meio da função de fitness (Equação 4.7). Uma nova geração é então composta por novos indivíduos a partir de indivíduos selecionados na população anterior por uma técnica de seleção chamada de roleta com estratégia elitista. Para gerar novos cromossomos são aplicados os operadores de *crossover* e mutação (taxas de 100% e 5% respectivamente). O crossover de ponto único

recombina dois indivíduos a partir de uma posição escolhida aleatóriamente (chamado ponto de corte), trocando os bits das strings a partir deste ponto. Durante a operação de mutação os pontos são escolhidos, de acordo com a probabilidade de mutação, e os valores destes bits são invertidos. A partir daí uma nova geração de indivíduos foi criada e uma nova iteração pode ser criada. O processo repete-se até um circuito adequado seja encontrado ou até um número pré-determinado de iterações seja alcançado.

Como resultado deste processo, foram obtidos (dentre outros não abordados neste trabalho) um controlador Kp e um controlador EHW, ambos de 4 bits, para pêndulo não linear. A tabela 4.1 mostra o comportamento de ambos.

Entrada	Saída Kp	Saída EHW
0000	0000	0000
0001	0001	0100
0010	0001	0011
0011	0010	0101
0100	0011	0010
0101	0011	0011
0110	0100	0001
0111	0100	0100
1000	1100	1111
1001	1100	0000
1010	1100	0000
1011	1101	1010
1100	1101	1101
1101	1110	1011
1110	1111	1110
1111	1111	0000

Tabela 4.1: Tabela verdade Kp e EHW

A figura 4.7 exibe um gráfico comportamental onde é possível observar melhor a característica não-linear do controlador evolutivo. Isso decorre da característica do algoritmo genético em otimizar funções não-lineares através de buscas conduzidas por um aprendizado experimental. O controlador proporcional, cuja função está representada pela cor vermelha no gráfico da figura 4.7 exibe um comportamento linear em resposta a entrada dos sinais de erro. Dessa forma, um tratamento linear é aplicado a um problema com comportamento não-linear, e isso ocorre devido às características intrínsecas dos controladores PID e suas variações (Equação 4.2, 4.3, 4.4 e 4.6). No controlador evolutivo, representado pela cor verde, observa-se uma função não linear em resposta ao sinal de erro. Esse comportamento é reflexo da evolução do

circuito controlador, que provê alterações adaptativas no projeto do circuito com base em suas experiências de controle frente ao problema proposto.

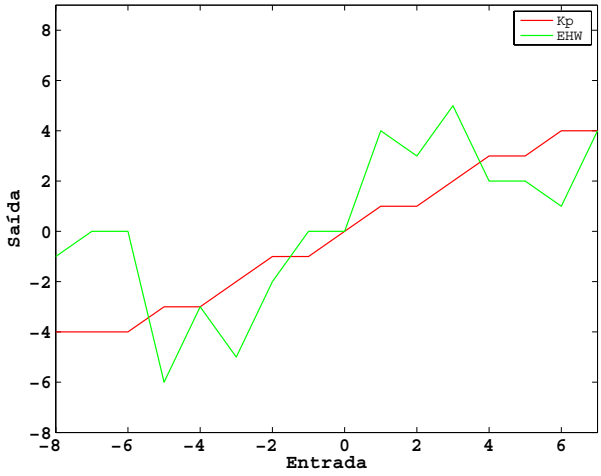

Figura 4.7: Comportamento dos controladores Proporcional e EHW

A técnica de Hardware Evolutivo foi vista nesse capítulo como uma abordagem alternativa ao problema de controle de sistemas dinâmicos. O controlador evolutivo obtido possui uma abordagem caixa preta, cuja função de controle foi conseguida através de um processo heurístico por meio de um AG. No capítulo a seguir serão mostrados detalhes da aplicação dos controladores e os resultados experimentais dos testes realizados em bancada.

Capítulo 5

APLICAÇÃO E RESULTADOS EXPERIMENTAIS

5.1 Implementação Física do Sistema de Controle

Como citado anteriormente, o objetivo deste trabalho é o de implementar fisicamente um modelo teórico de controle de pêndulo em malha fechada, obtido através da técnica de *hardware* evolutivo e sintonia de controlador proporcional por algoritmo genético (Campos, 2007).

Para criar um modelo físico com base no modelo simulado primeiramente é necessário estudar o modelo proveniente de simulação a fim de obter-se informações que serão usadas na escolha de quais componentes serão necessários, e conseqüentemente utilizados, no projeto físico do sistema de controle experimental. A figura 5.1 exibe um diagrama geral de todo o sistema experimental criado com base no modelo teórico proposto por Campos (2007).

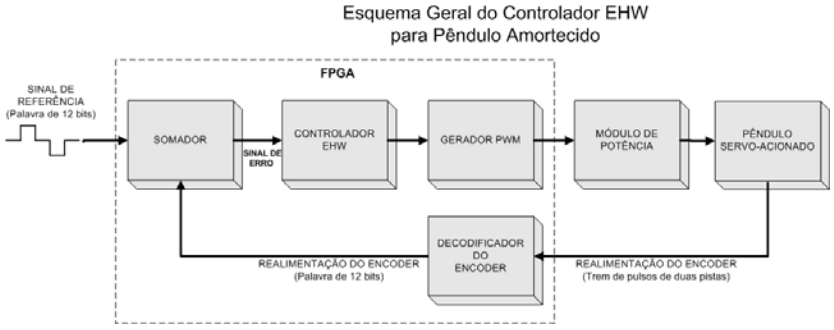

Figura 5.1: Diagrama geral do controlador

5.1.1 Pêndulo Servo Acionado

O objeto alvo do controle é um pêndulo mecânico desenvolvido no Laboratório de Sistemas Modulares Robóticos (LSMR) da FEEC - Unicamp e possui uma haste de 28,5 cm de comprimento e um conjunto de engrenagens de redução acionadas por um motor CC alimentado com 24 volts (figura 5.2). A este motor é acoplado um *encoder* óptico incremental marca HP modelo HEDS-6310 formando assim o sistema servo-motor. Este *encoder* possui 1000 pulsos de resolução e dois canais defasados em 90º cada, para informação do sentido de rotação.

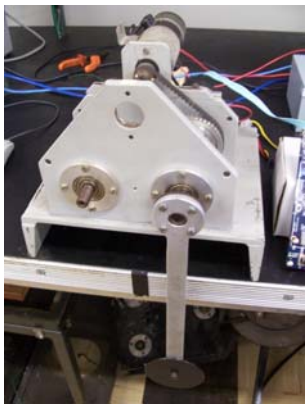

Figura 5.2: Pêndulo servo-acionado

A haste do pêndulo move-se no sentido horário e anti-horário dependendo de como a polarização é aplicada nos terminais do motor CC. Como se trata de um motor de corrente contínua, a velocidade de rotação e conseqüentemente de deslocamento angular da haste são dependentes, dado o modelo dinâmico do pêndulo, do nível de corrente aplicada aos terminais do motor (no caso deste sistema, níveis entre 0 e 24 volts).

5.1.2 Alimentação do Sistema

A responsabilidade do suprimento de energia para o sistema foi dividida entre duas fontes simétricas de tensão contínua. A primeira fonte, uma fonte variável da marca ICEL modelo PS5000D (figura 5.3) com saída variável de 0 a 30 Volts e fornecimento máximo de 6 Amperes de corrente contínua, foi usada para alimentar o circuito de potência acionador do pêndulo.

A segunda fonte é uma fonte simétrica chaveada padrão ATX usada na alimentação de microcomputadores arquitetura IBMPC e compatíveis (figura 5.4). Essa fonte é usada na alimentação do circuito analógico de bufferização que interconecta os vários sinais de controle do sistema.

Figura 5.3: Fonte do acionador do pêndulo

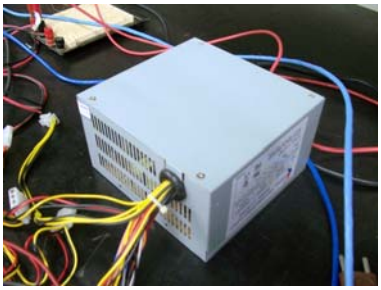

Figura 5.4: Fonte do circuito de interfaceamento

5.1.3 Circuito de Interfaceamento

Um dos problemas que surgiram durante a construção do modelo experimental de bancada foi que os sinais de controle que circulavam pelo sistema não eram totalmente compatíveis com níveis de sinais TTL (Transistor Transistor Logic) utilizados, por exemplo, pelo encoder do pêndulo e pelo PC gerador de sinal de referência. O padrão TTL utiliza 0 Volts para representar o dígito binário 0 e 5 Volts para representar o dígito 1. Esse problema deve-se ao fato de que, devido a sua engenharia interna, o circuito FPGA opera com níveis de 0 volts para o dígito 0 e 3,3 volts para o dígito 1 nas suas portas de E/S. Além disso, o circuito de potência que recebe os sinais PWM de controle e que a princípio deveria operar com 5 volts em nível lógico alto (dígito 1) só obtinha resposta satisfatória com 12 volts em suas entradas.

Devido a essa necessidade de conversão entre os níveis elétricos dos sinais, foi necessário projetar um circuito analógico de bufferização de sinais para interfacear os dispositivos eletricamente incompatíveis (figura 5.5). Esse circuito de interfaceamento elétrico utiliza 3 CI's da família TTL modelo 74LS07 que possui internamente 6 *buffers* com saída em coletor aberto. As saídas possibilitam a ligação em paralelo de resistores de *pull-up* para obter os níveis de tensão desejados na realização das conversões de 5 para 3,3 e 12 volts e também de 3,3 para 12 Volts (conversão necessária para o sinal de PWM enviado pelo FPGA). Além dos *buffers*,

este módulo também possui um CI 7404 que possui 6 inversores sendo um deles destinado ao interfaceamento do sinal de PWM com o acionador de potência do pêndulo.

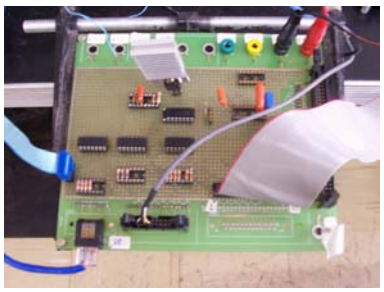

Figura 5.5: Circuito de interface

O circuito de *buffers* utiliza 12, 5 e 3,3 Volts. Os 12 volts são convertidos para 5 volts por um CI regulador de tensão modelo 7805 e entregue aos pinos de alimentação dos *buffers* 7407.

5.1.4 Kit Fpga de Desenvolvimento

Muitos dos componentes do sistema controlador do pêndulo são circuitos digitais. Sendo assim, a implementação desses componentes torna-se possível e até mesmo mais viável (do ponto de vista da velocidade de projeto e testes) em circuitos lógicos reconfiguráveis, como é o caso dos FPGA's.

Para sintetizar os circuitos digitais necessários ao projeto foi utilizado um Kit de desenvolvimento da empresa Altera *Corporation*, modelo DK-CYCII-2C20N-0A (figura 5.6), que contém um *chip* FPGA família Cyclone II modelo EP2C20F484C7N de 18.752 elementos lógicos 236.616 bits de memória RAM e 315 pinos de E/S (Altera, 2007).

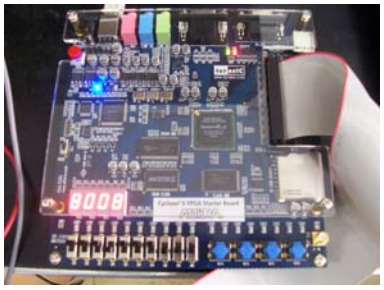

Figura 5.6: Kit de Desenvolvimento Altera

Além do chip FPGA o kit possui também outros componentes como um encoder de

áudio de 24 bits, uma memória *sram* de 512 KB, uma memória *sdram* de 8 megabytes, circuito de vídeo VGA, interface serial padrão RS232, memória *flash* de 4 megabytes, 4 displays de 7 segmentos, 10 chaves de dois estados, um slot para cartão de memória SD e 2 *slots* de 40 pinos cada para conexão com dispositivos externos. A programação do *kit* é realizada através de uma interface de comunicação USB que pode ser ligada a um PC *desktop* ou a um *notebook* que contenha um programa de projetos e simulações da Altera, como o Quartus II (Altera, 2008).

5.1.5 Sistema de Geração de Sinal de Referência

A resposta do sistema do pêndulo é baseada no sinal de referência aplicado a entrada do controlador, esse sinal é uma palavra binária de 12 bits que indica ao controlador para qual posição a haste do pêndulo deve se deslocar e pode variar entre os valores decimais de 0 a 4000 (figura 5.7).

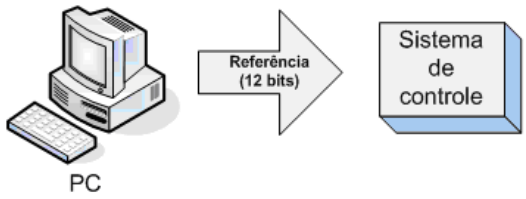

Figura 5.7: Esquema do gerador de referência

Devido a flexibilidade e disponibilidade o gerador de referência adotado foi um micro-computador arquitetura IBMPC equipado com um processador Pentium 3 e 128 megabytes de memória RAM (figura 5.8). O sistema operacional usado foi o MS-DOS versão 6.22 devido as suas características de execução de programas em tempo real e a relativa facilidade de uso e acesso à porta paralela.

Figura 5.8: PC gerador de sinal de referência

Para gerar o sinal de referência foi criado um pequeno software em linguagem C que

disponibiliza ao usuário as operações mostradas na interface textual da figura 5.9.

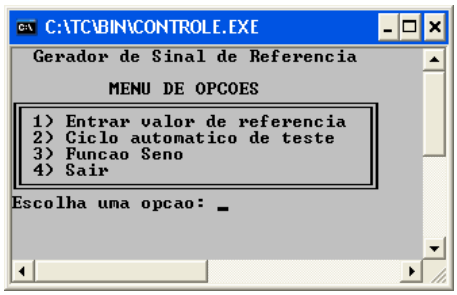

Figura 5.9: Interface do Sofware

É importante salientar que o computador é responsável somente pela geração do sinal de referência, não recebendo nenhum sinal de retorno do sistema de controle, ou seja, o PC não exerce nenhuma ação direta no controle do pêndulo, sendo essa uma responsabilidade única do controlador sintetizado no FPGA.

5.1.6 Módulo de Potência

Para realizar o acionamento do pêndulo, em resposta ao sinal PWM enviado pelo controlador, foi necessário utilizar um circuito de potência que trabalhasse com freqüências de chaveamento relativamente altas e fornecimento de corrente elétrica suficiente à alimentação do motor CC que movimenta a haste. Inicialmente, um circuito de acionamento foi projetado para a tarefa de acionamento, mas devido a problemas com um dos componentes e ao tempo limitado para conclusão de todo o sistema optou-se por utilizar um módulo previamente desenvolvido.

O circuito de acionamento é um circuito usado nos laboratórios de graduação da FEEC e contém um *chip* ponte H modelo IRAMXUP60A (Pomilio, 2008) com entradas de acionamento opto-isoladas, como pode ser observado no diagrama da figura 5.10. Neste circuito, opto-acopladores garantem um isolamento elétrico entre o módulo de acionamento de potência do pêndulo e os demais circuitos do sistema.

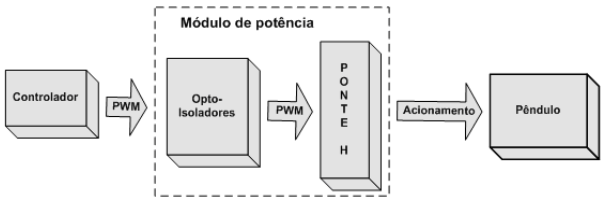

Figura 5.10: Diagrama do módulo de potência

5.2 Síntese do Sistema em FPGA

A implementação do controlador digital e também dos demais módulos do projeto foi realizada utilizando diagramação no ambiente de desenvolvimento Altera Quartus II. Cada um dos blocos do diagrama corresponde a um circuito digital do sistema de controle do pêndulo e podem ser observados na figura A.1 e figura A.2. Alguns blocos do digrama foram implementados utilizando linguagem VHDL e outros utilizando componentes da biblioteca do próprio QuartusII, como por exemplo portas lógicas e CIs (Circuitos Integrados). O desenvolvimento em ambientes EDA como o Quartus provê a praticidade da realização de testes dos circuitos tanto em ambiente de simulação quanto em ambientes físicos reais (com sinterização em FPGA). Todos os blocos foram testados fisicamente através dos recursos disponibilizados pelo kit de desenvolvimento Altera tais como leds, chaves seletoras, botões de pressão, displays e pinos de I/O.

O processo de criação dos blocos consiste basicamente em desenvolvê-los utilizando o ambiente gráfico do QuartusII. Posteriormente é realizada a compilação para verificação de possíveis erros, o mapeamento para os pinos de I/O do FPGA e por fim a gravação do circuito no chip. Optou-se em desenvolver os blocos em projetos separados e, depois de prontos, uni-los em um único projeto. Isso implica em maior facilidade principalmente na etapa de testes de cada módulo.

5.2.1 Bloco de Interface com o PC

Como citado anteriormente, um microcomputador arquitetura IBM/PC foi utilizado para a geração de sinal através de uma porta de comunicação paralela. A interface de comunicação paralela dos PCs é na verdade dividida em 3 outras portas chamadas de portas de dados (saída de 8 bits), status (entrada de 5 bits) e controle (saída de 4 bits) (figura 5.11).

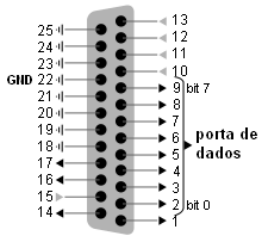

Figura 5.11: Interface paralela IBM/PC

Apenas a porta de dados foi utilizada no projeto para enviar o sinal de referência de 12

bits na forma de três segmentos de 4 bits cada, ou seja, a palavra de 12 bits foi construída através da concatenação de palavras menores de 4 bits. Para que isso fosse possível, foi necessária a criação de um bloco dentro do *chip* FPGA para realizar a concatenação das palavras menores, e este corresponde ao bloco 1 da figura A.1.

O bloco consiste internamente de três registradores de 4 bits e um registrador de 12 bits, todos circuitos assíncronos (figura 5.12). Os três primeiros registradores de 4 bits armazenam as componentes da palavra do sinal de referência e o quarto registrador é responsável por habilitar o registrador de 12 bits que recebe a palavra completa. O *nibble* menos significativo da porta de dados envia as palavras de 4 bits enquanto *nibble* mais significativo seleciona qual dos registradores estará habilitado a receber o dado corrente. Esse processo garante que mesmo utilizando apenas os 8 bits da interface paralela do PC, palavras de 12 bits possam ser enviadas para o sistema de controle do pêndulo.

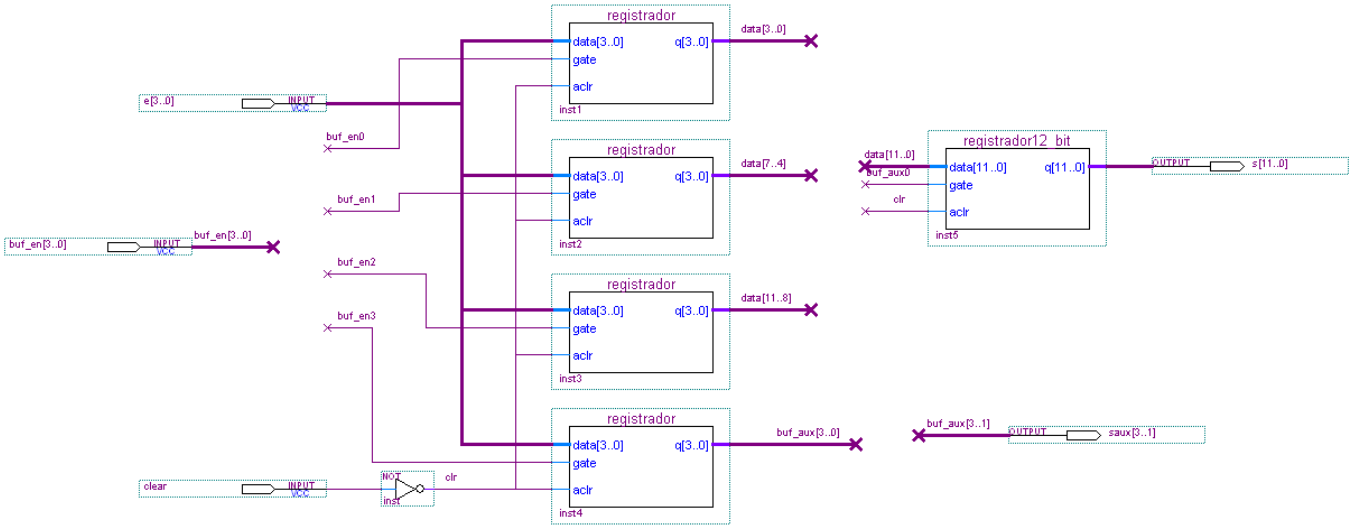

Figura 5.12: Bloco de interface com PC (expandido)

O gerenciamento de seleção dos registradores e do envio dos bits a suas portas de entrada fica a cargo do software de geração do sinal de referência, que é executado no PC.

5.2.2 Bloco Somador

O bloco somador (bloco 2 da figura A.1) foi implementado para realizar a subtração do sinal de referência com o sinal de posição do pêndulo, ambos de 12 bits. Como resultado da subtração tem-se o sinal de erro também de 12 bits ($r(t) - y(t) = e(t)$). Este somador realiza operações em complemento de dois, o que permite a geração de valores negativos em sua saída.

A construção do bloco somador foi realizada utilizando código VHDL através do recurso MegaWizard Plug-in Manager disponível no ambiente de desenvolvimento Quartus II, esse recurso permite a criação de componentes de forma rápida através da passagem de alguns parâmetros tais como número de bits das entradas, tipos de operação etc.

5.2.3 Blocos de Ajuste de Referência

O objetivo destes blocos é realizar o ajuste da referência da posição do pêndulo adequando-o ao sistema de controle proposto. Esse blocos são compostos de uma lógica interna implementada em VHDL e podem ser observados no diagrama da figura A.1 (Blocos 3 e 4) e na figura A.2 (Bloco 14).

5.2.4 Bloco de Saturação do Sinal de Erro

O sistema de controle do pêndulo trabalha com um sinal de referência de 12 bits sendo o sinal de posição que retorna do pêndulo também de 12 bits. O faixa de valores especificada para o sistema dentro desses bits vai de 0 a 4000 e consequentemente o maior valor de erro que o bloco somador irá sinalizar será de 4000, embora isso não ocorra na prática.

Os controladores Kp e EHW possuem uma entrada de 4 bits para receber o sinal de erro proveniente do somador. Dessa forma, os controladores só conseguem "enxergar" uma faixa de erro de -8 a 7, o que representa apenas 16 valores (2^4) , contra os 4000 que o erro pode assumir. Para adequar o erro a faixa de valores que os controladores podem operar, foi criado em VHDL um bloco de saturação (bloco 5 figura A.1). Esse bloco recebe 12 bits de entrada e fornece em sua saída os quatro bits compatíveis com os controladores. O bloco mantém saída fixa em -8 para valores de erro de entrada inferiores a tal e saída fixa em 7 para erros superiores a este valor. Só haverá variações na saída do circuito de saturação quando o erro estiver dentro desta faixa, neste caso o valor de saída será o mesmo valor da entrada.

5.2.5 Bloco Decodificador do Encoder

Este bloco é responsável por adquirir e condicionar o sinal proveniente do *encoder* HP que está acoplado ao eixo do pêndulo (bloco 6 figura A.1). O bloco decodificador é um circuito síncrono que utiliza um *clock* de 27 Mhz fornecido por um dos osciladores internos do kit de desenvolvimento Altera. O sinal do *encoder* é entregue na forma de duas ondas quadradas

chamadas de pistas. As pistas 1 e 2 são defasadas em 90 graus entre si (figura 5.13).

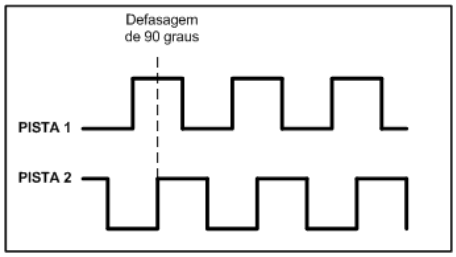

Figura 5.13: Sinais provenientes do encoder

O bloco decodificador recebe as duas pistas convertendo-as em uma palavra binária de 12 bits que representa a posição da haste do pêndulo. Internamente, o bloco implementa um contador circular de 0 a 4000 e uma máquina de 4 estados que, com base nos sinais fornecidos pelo encoder, define o sentido de rotação. Os estados em que a máquina pode se encontrar, baseado nos sinais do encoder, podem ser vistos na figura 5.14.

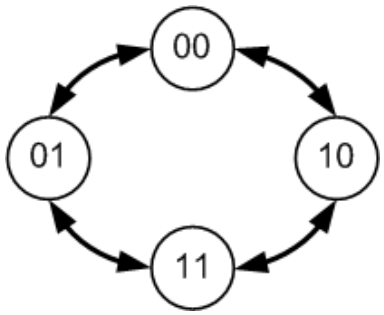

Figura 5.14: Estados da máquina do decodificador

Por exemplo, transições do estado 00 para o estado 10 indicam que haste está se deslocando em um sentido, sendo que transições do estado 00 para 11 indicam que o deslocamento é no sentido contrário. Isso possibilita o incremento ou decremento do contador circular informando a posição correta da haste.

5.2.6 Bloco Seletor de Sinais

O bloco seletor número 7 do apêndice A.1 foi criado como recurso para observação das informações de posição de pêndulo e também do erro, em tempo real. Este bloco consiste em um circuito que recebe em suas entradas os 12 bits da posição da haste e também os 12 bits do sinal de erro proveniente do bloco somador e realiza o chaveamento entre eles, enviando

um ou outro para os blocos responsáveis por escrever nos *displays* de 7 segmentos do kit de desenvolvimento Altera. Este bloco funciona com um multiplexador dos sinais de posição e de erro onde o usuário pode escolher manualmente a visualização do sinal desejado através da mudança de uma chave de dois estados situada também no kit.

Tanto a posição quando o erro são exibidos em 3 *displays* utilizando numeração em base hexadecimal, sendo que, no caso da escolha pelo sinal de erro, um quarto *display* é utilizado para exibir o sinal "-", quando o erro for negativo. A figura 5.15 ilustra o funcionamento básico do bloco seletor.

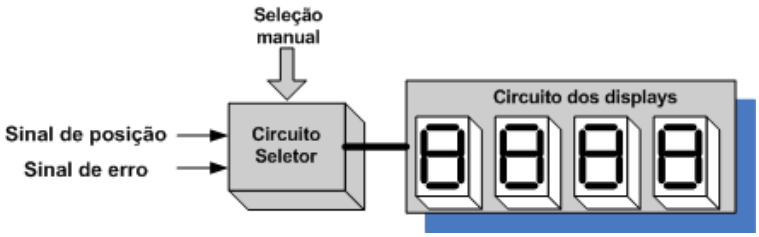

Figura 5.15: Esquema do bloco seletor

5.2.7 Bloco Decodificador do Display

O bloco 12 da figuraA.1 representa o circuito digital que decodifica números binários de 4 bits para sua representação hexadecimal em um *display* de 7 segmentos.

Os *displays* são usados para mostrar visualmente os sinais pertinentes ao sistema de controle do pêndulo tais como posição da haste e o sinal de erro. No projeto do sistema de controle são utilizados 3 blocos decodificadores com suas saídas ligadas a 3 dos 4 *displays* do kit de desenvolvimento Altera e suas entradas ligadas ao bloco seletor de sinais.

O bloco decodificador do *display* é um circuito digital assíncrono descrito em linguagem VHDL.

5.2.8 Bloco do Controlador

O bloco do controlador (bloco 10 figura A.2) é o circuito do controlador digital do pêndulo. Foram implementados em VHDL dois blocos neste projeto (figura 5.16), um para o controlador EHW e outro para o controlador Proporcional, ambos circuitos digitais combinacionais de 4 bits de entrada e 4 bits de saída. Os blocos são trocados e o projeto é novamente

recompilado quando deseja-se realizar experimentos com qualquer um dos dois.

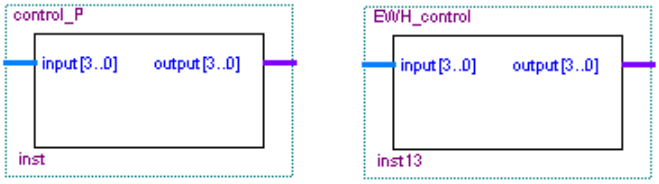

Figura 5.16: Blocos de controle propocional e EHW

5.2.9 Bloco Tradutor PWM

O bloco de controle (tanto EHW quando Kp) tem suas entradas ligadas a saída do bloco saturador recebendo, dessa forma, sinais de erro no intervalo de -8 a 7 provenientes da planta. As saídas do controlador são conectadas a um bloco que realiza a tradução da palavra binária entregue (também variando de -8 a 7) para o bloco gerador de sinal PWM.

O circuito controlador EHW / Kp determina o nível e polaridade da energia elétrica entregue ao motor do pêndulo através de uma palavra binária de 4 bits. O circuito gerador do sinal de PWM implementado no projeto não possui compatibilidade direta com os sinais entregues pelo circuito de controle e isso é um problema, pois se o controlador indica um nível x para o sinal PWM o gerador pode responder com um nível y, comprometendo o processo de controle.

Com o propósito de resolver a incompatibilidade entre os dois blocos foi desenvolvido um bloco intermediário denominado bloco tradutor pwm (bloco 11 figura A.2). Esse bloco implementa um circuito de 4 entradas e 4 saídas que recebe a palavra do controlador e a traduz para que o gerador PWM possa entendê-la.

A tabela 5.1 exibe as respostas do tradutor frente as entradas fornecidas pelo controlador e também o nível percentual do sinal PWM entregue ao motor do pêndulo pelo circuito gerador. O sinal "+" indica polarização de giro do motor no sentido anti-horário e o sinal de "-" no sentido horário. Em resumo essa tabela exibe em suas colunas valores que representam o comportamento dos controladores para todas as entradas possíveis entregues pelo circuito somador que calcula o erro.

Valores Decimais	Saída do Controlador	Saída do Tradutor	Nível PWM
0	0000	0111	0%
1	0001	0110	+ 12,5%
2	0010	0101	+ 25%
3	0011	0100	+ 37,5%
4	0100	0011	+ 50%
5	0101	0010	+ 62,5%
6	0110	0001	+ 75%
7	0111	0000	+ 87,5%
-8	1000	1111	- 100%
-7	1001	1110	- 87,5%
-6	1010	1101	- 75%
-5	1011	1100	- 62,5%
-4	1100	1011	- 50%
-3	1101	1010	- 37,5%
-2	1110	1001	- 25%
-1	1111	1000	- 12,5%

Tabela 5.1: Sinais do bloco tradutor e PWM

5.2.10 Bloco Gerador de Sinal PWM

Este bloco (bloco 13 figura A.2) representa o circuito gerador do sinal de PWM que pode produzir sinais de 16 níveis diferentes, correspondentes às 16 palavras de 4 bits que o controlador é capaz de gerar, visando o controle da energia enviada ao pêndulo. A codificação do sinal de controle PWM foi escolhida devido ao fato do motor do pêndulo ser um motor CC, sendo essa codificação normalmente usada no controle dessa classe de motores. A técnica de PWM é utilizada para variar o valor da transferência de potência entregue a uma carga sem as perdas ocorridas normalmente devido à queda de tensão por recursos resistivos.

Na implementação deste bloco foram usados dois outros blocos da biblioteca do ambiente de desenvolvimento Quartus II. O primeiro bloco representa um circuito integrado 74161 que é um contador síncrono de 4 bits. O segundo bloco é o do CI 7485 que é um comparador de módulo 4.

A configuração adotada no diagrama da figura 5.17 caracteriza um circuito síncrono com uma entrada de 50 MHz e divisões de *clock* que permitem que seja gerado em sua saída um sinal PWM de aproximadamente 6 Khz e com níveis determinados pelas entradas b0 a b3, o que possibilita a geração dos 16 níveis necessários ao sistema.

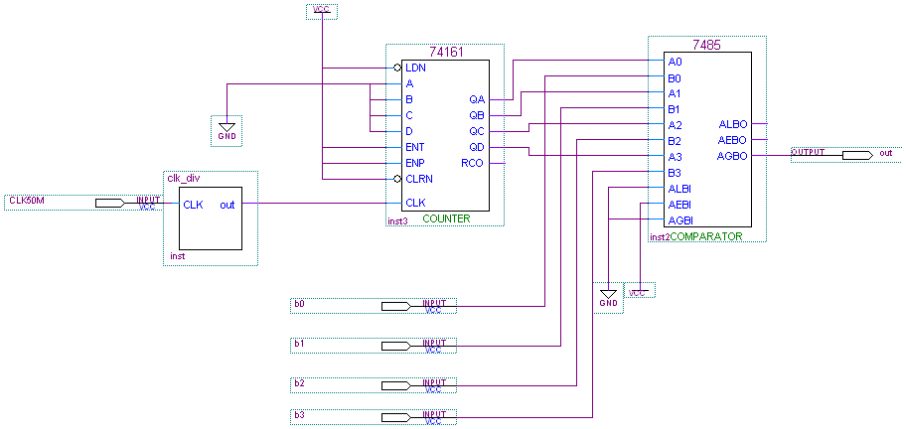

Figura 5.17: Diagrama do bloco gerador de sinal PWM (expandido)

5.3 Panorama Geral do Sistema

De uma forma geral é possível resumir o funcionamento de todo o sistema da seguinte forma: Um sinal de referência é inserido no sistema por meio de um programa sendo executado em um computador externo. Este sinal indica para qual posição a haste do pêndulo deve se deslocar. O sistema controlador, em resposta ao sinal de referência, verifica se a posição atual da haste é a mesma indicada pela referência, caso não seja ações são tomadas no sentido de se atingir a referência. Isso é feito através do controle de energia enviada ao motor do pêndulo realizada na modulação por largura de pulso (PWM), que alterna em função das "ordens" dadas pelo controlador Kp ou EHW. Além do nível de energia, a polaridade dos sinal também é controlada para movimentar a haste no sentido correto (horário e anti-horário). Todo o controle é realizado em tempo real através dos circuitos digitais sintetizados no *chip* FPGA, e dessa maneira não existem *softwares* sendo executados dentro do sistema de controle do pêndulo. Um diagrama contendo os componente de hardware de todo o sistema pode ser visto na figura 5.18.

É importante citar que, por se tratar de um trabalho prático, problemas envolvendo as conexões externas ao FPGA foram enfrentados no decorrer de toda a implementação, tais como maus contatos, problemas de alimentação, incompatibilidade elétrica, sincronização, entre outros. Mas, de uma forma geral, o acoplamento e o funcionamento de todos os módulos que compõem o sistema de controle atuaram de forma satisfatória na realização dos testes entre os dois controladores sintetizados individualmente no FPGA. A bancada experimental onde todo o sistema foi montado pode ser vista na figura 5.19.

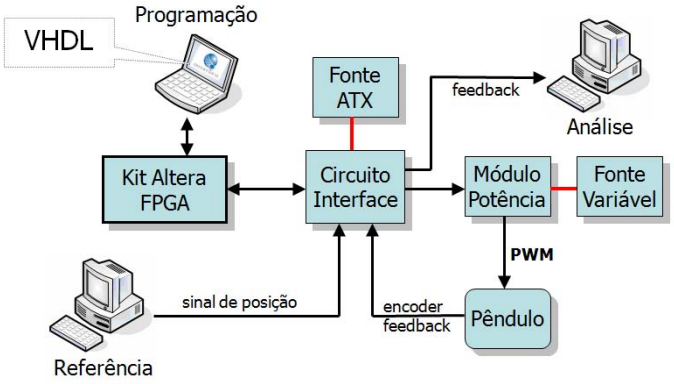

Figura 5.18: Diagrama dos componentes de hardware

Figura 5.19: Bancada experimental

5.4 Sistema de Aquisição de Dados

A etapa de aquisição de dados objetivou a obtenção de vetores de informação contendo os pontos que tornaram possível a construção de gráficos de análise comportamental dos controladores Kp e EHW para os experimentos.

O sistema usado na aquisição foi composto por dois computadores (IBM/PC) ligados em rede através de interfaces ethernet com velocidade de 100 Mbps. O protocolo usado na comunicação entre os dois computadores foi o TCP/IP. Neste sistema, uma abordagem mestre/escravo é utilizada onde um dos computadores é considerado mestre ou *host* executando um sistema operacional Windows XP e o software MATLAB / SIMULINK. O SIMULINK disponibiliza uma interface gráfica com blocos dedicados às mais diversas tarefas incluindo blocos usados

na aquisição e tratamento dos dados disponibilizados pelo sistema de controle do pêndulo. O segundo computador é chamado escravo ou *target* e executa um sistema operacional de tempo real dedicado, cuja responsabilidade é a de realizar a aquisição dos sinais de posicionamento do pêndulo e repassa-los ao host. A esse sistema de aquisição é dado o nome de XPC TARGET e está ilustrado na figura 5.20.

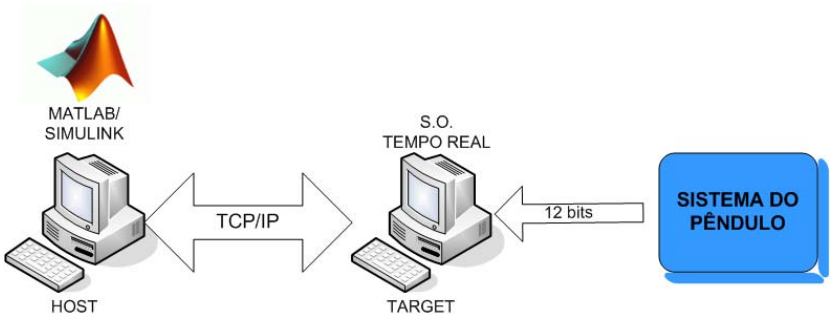

Figura 5.20: Sistema de aquisição de dados XPC*target*

Para a aquisição do sinal de posição do pêndulo que é composto por 12 bits, foi utilizada uma placa de aquisição da *National Semiconductor* acoplada ao barramento PCI do computador *target*. Os *drivers* de controle da placa são disponibilizados no próprio MATLAB / SIMULINK e adicionados ao sistema operacional do PC *target*. A taxa de amostragem do sinal posicional do *encoder* do pêndulo foi de 1Khz.

5.5 Dados Obtidos

Para os testes e geração dos gráficos foram utilizados dois sinais distintos de referência, tanto para o controlador Kp quando para o controlador EHW. O primeiro sinal de referência consistiu em uma onda quadrada semelhante à usada previamente nos modelos simulados por Campos (2007). O resultado do teste da implementação em laboratório para o controlador Kp pode ser observado na figura 5.21 sendo a figura 5.22 referente ao mesmo teste aplicado ao controlador EHW.

Os gráficos de teste entre os controladores exibem um melhor resultado em favor do controlador EHW, frente ao controlador Kp, já que é possível notar no controlador evolutivo um *overshoot* e um erro de regime menos acentuados. Esta observação fica mais evidente sobrepondo às duas funções, como mostrado na figura 5.23, onde é possível observar os resultados das diferentes funções de controle.

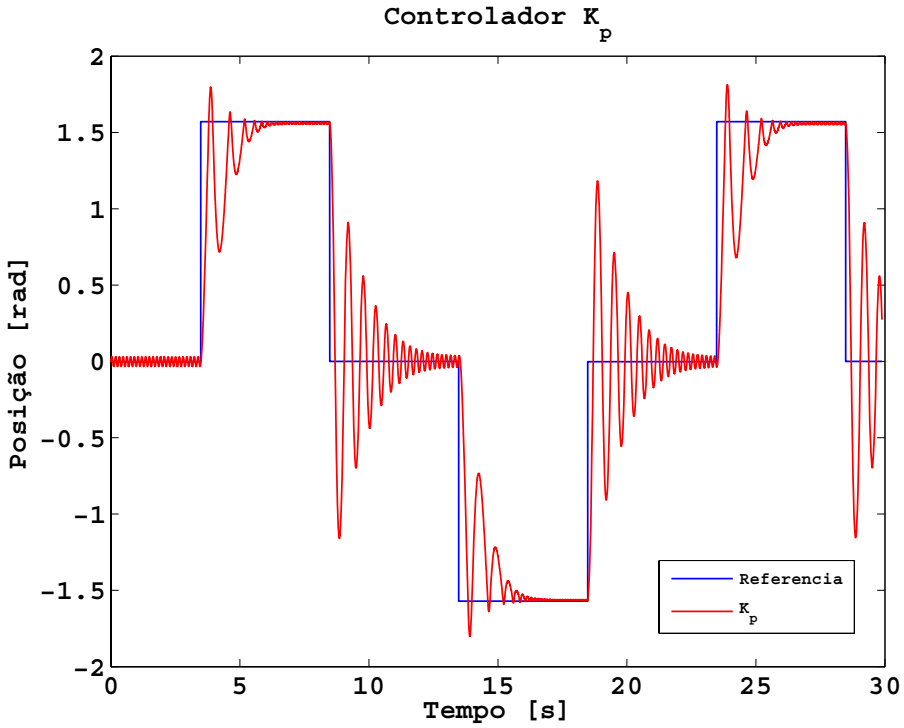

Figura 5.21: Resposta temporal do sinal de posiçao - kp

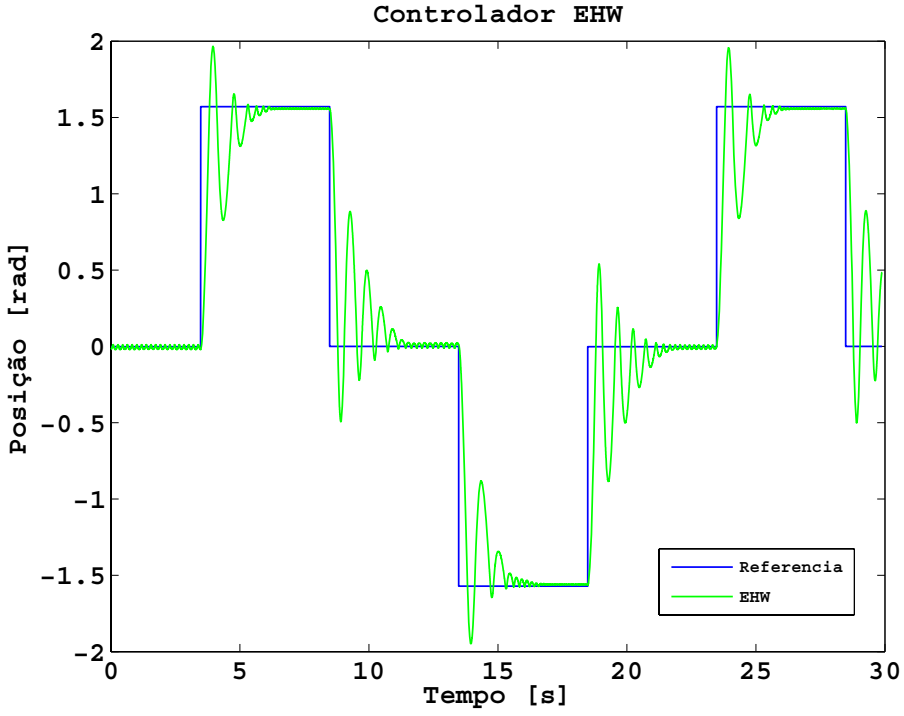

Figura 5.22: Resposta temporal do sinal de posiçao - EHW

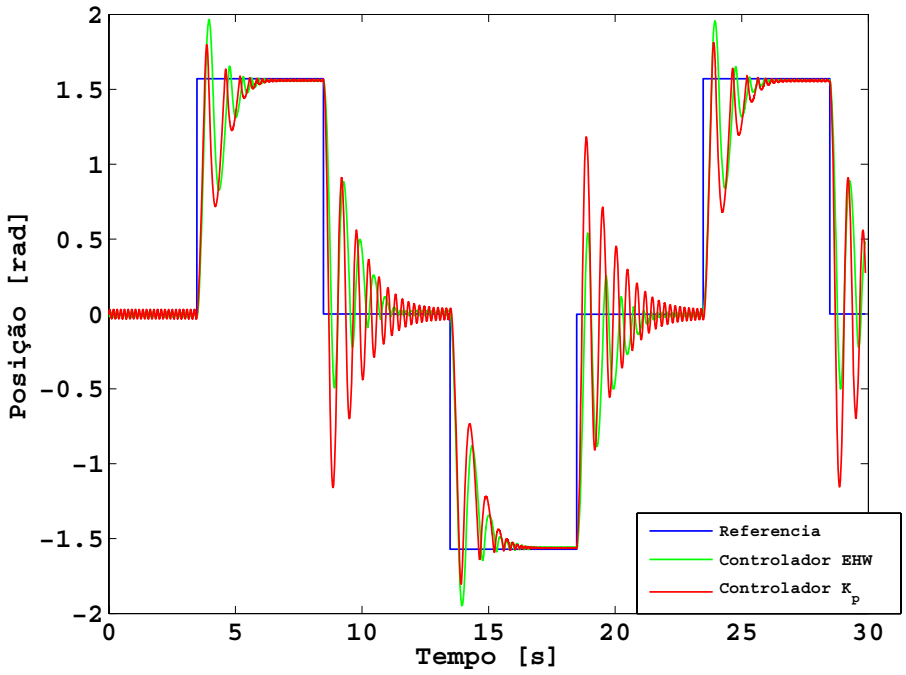

Figura 5.23: Resposta temporal do sinal de posição - Kp e EHW

Os gráficos das figuras 5.24, 5.25 e 5.26 exibem respectivamente os sinais de erro dos controladores Kp, EHW e a sobreposição de ambos.

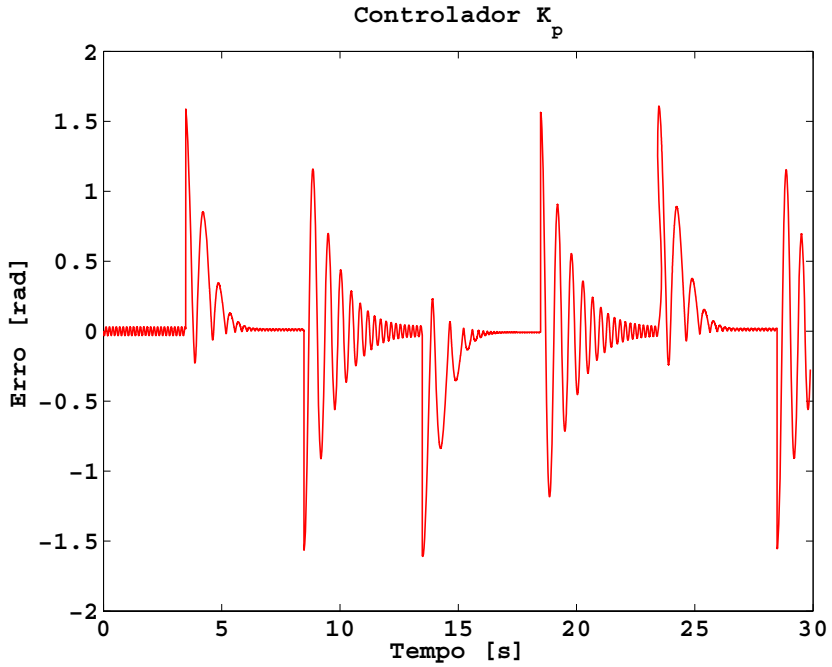

Figura 5.24: Resposta temporal do sinal de erro - Kp

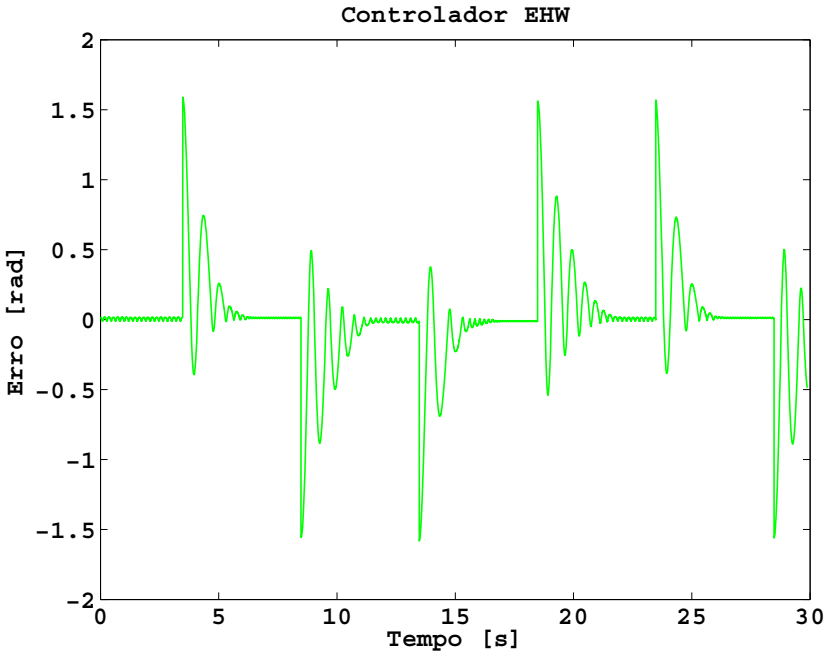

Figura 5.25: Resposta temporal do sinal de erro - ehw

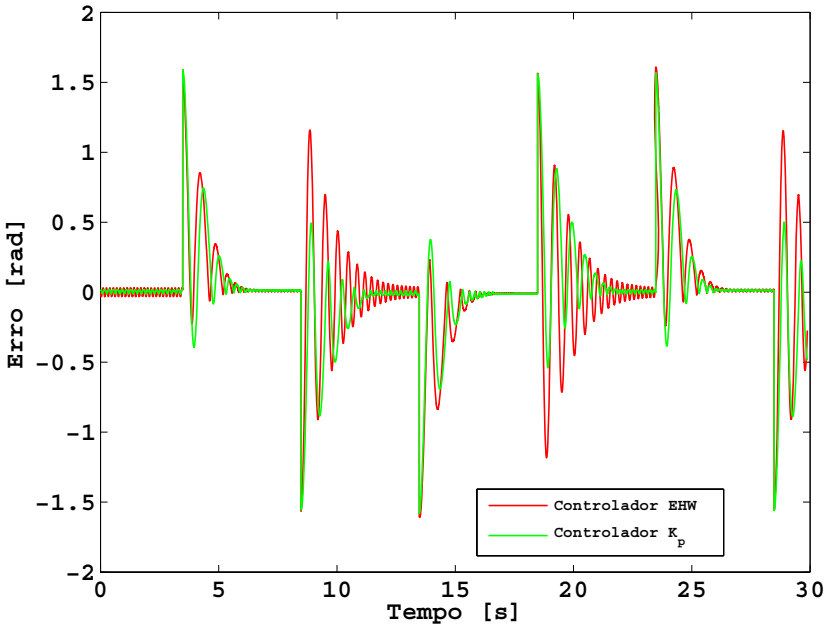

Figura 5.26: Resposta temporal do sinal de erro - Kp e ehw

O segundo sinal de referência adotado nos testes foi uma função seno definida também dentro dos limites posicionais de 1000 a -1000. Os resultados podem ser observados na figuras 5.27, 5.28, 5.29.

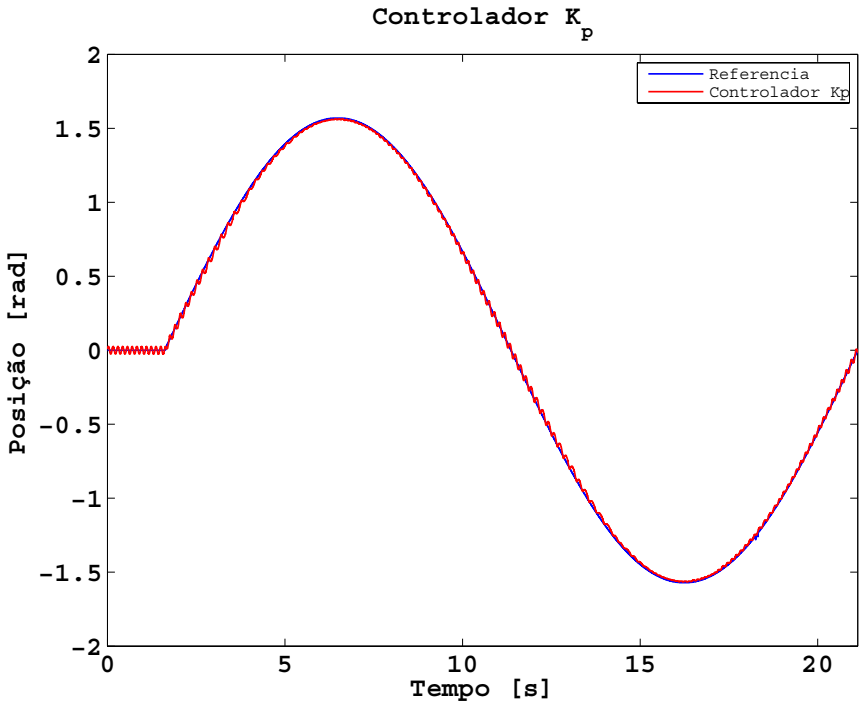

Figura 5.27: Resposta temporal do sinal de posição em função seno - Kp

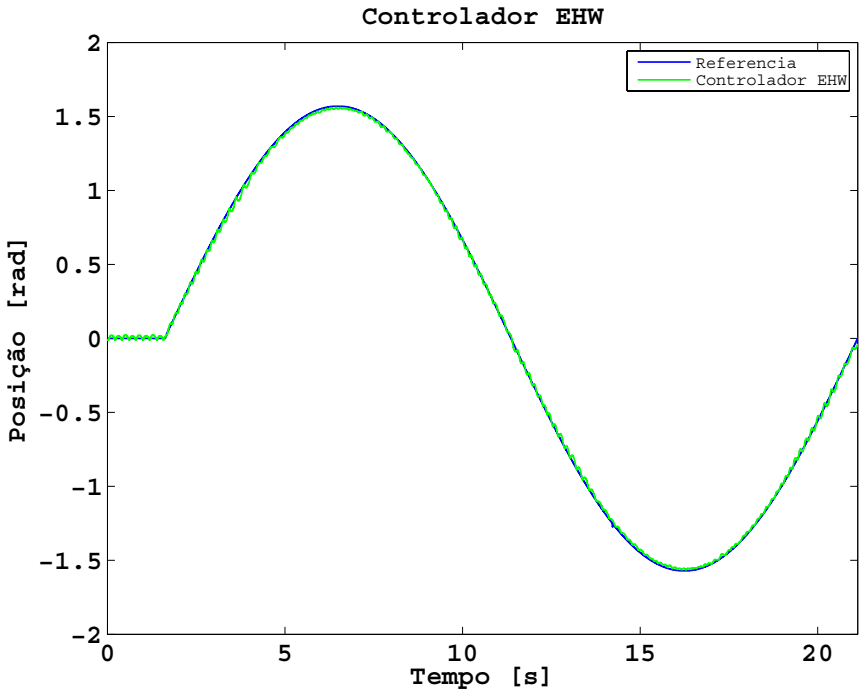

Figura 5.28: Resposta temporal do sinal de posição em função seno - EHW

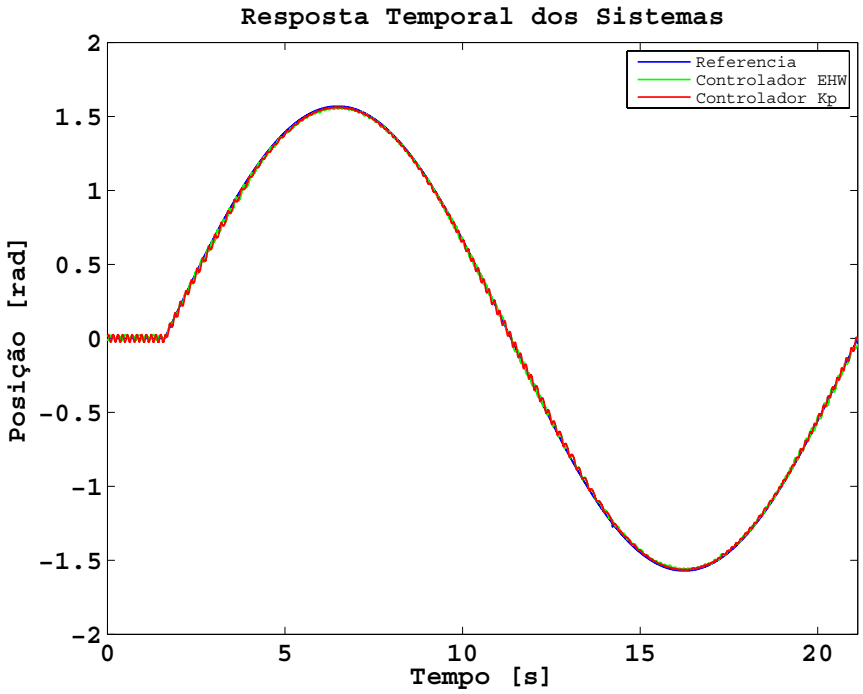

Figura 5.29: Resposta temporal do sinal de posição em função seno - EHW e Kp

Os gráficos das figuras 5.30, 5.31 e 5.32 exibem respectivamente os sinais de erro dos controladores Kp, EHW e a sobreposição de ambos para a referência senoidal.

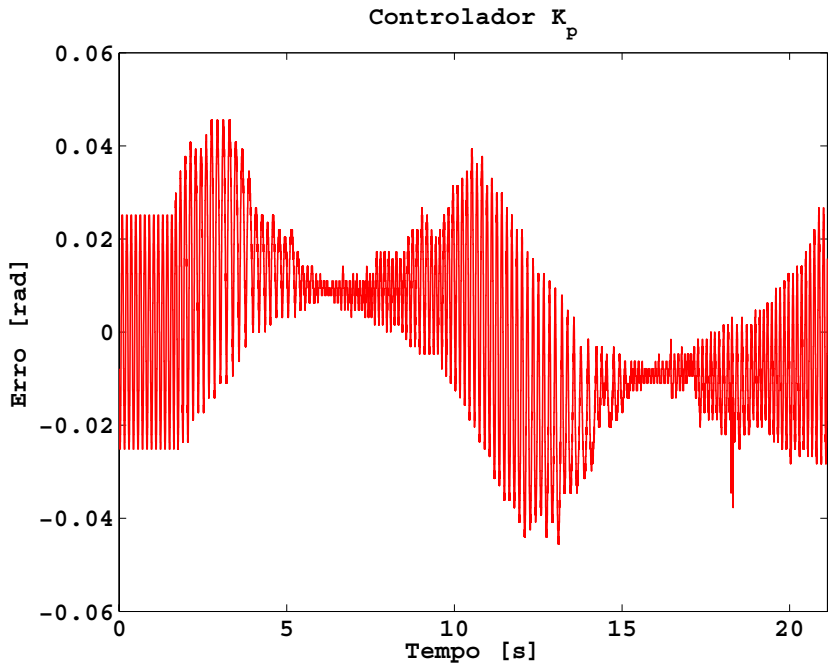

Figura 5.30: Resposta temporal do sinal de erro senoidal - Kp

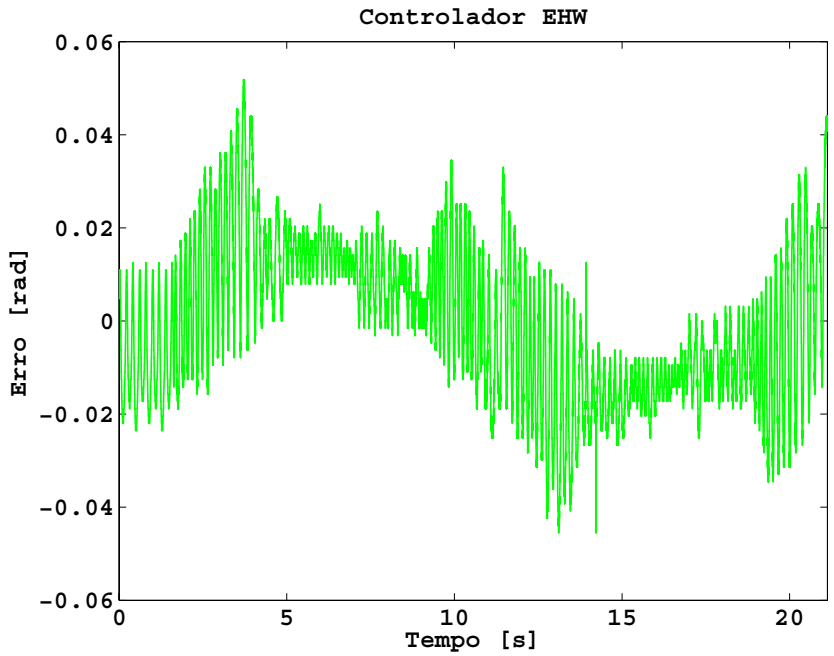

Figura 5.31: Resposta temporal do sinal de erro senoidal - EHW

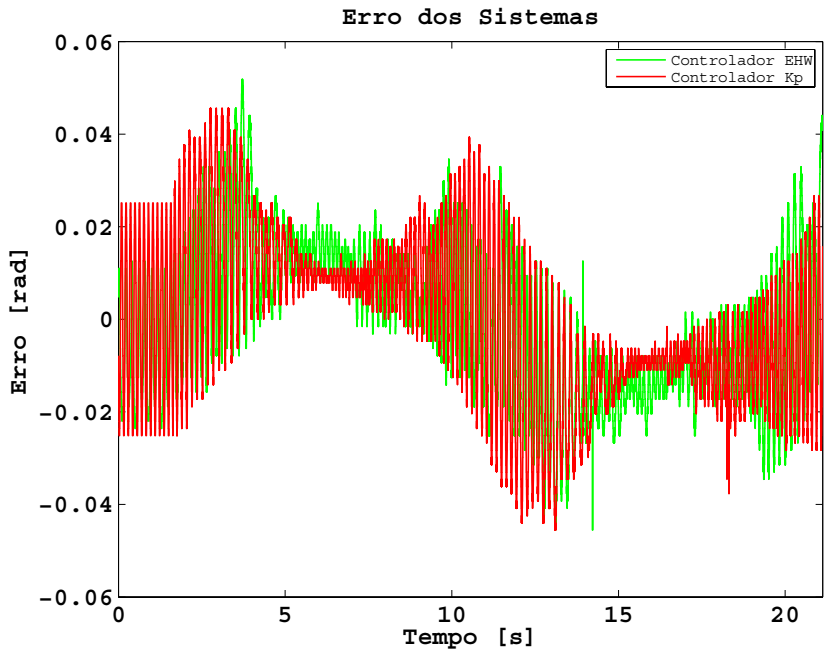

Figura 5.32: Resposta temporal do sinal de erro senoidal - EHW e Kp

5.6 Cálculo do Erro Médio Quadrático

Como critério comparativo entre o desempenho do controlador Kp e o controlador EHW foi realizado o cálculo do erro médio quadrático (RMS) para as duas funções aplicadas nos

testes (quadrada e seno) segundo a equação 5.1.

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \tag{5.1}$$

A tabelas 5.2 e 5.3 demonstram que, tanto para a referência onda quadrada quanto para o seno, o controlador EHW obteve um valor RMS menos acentuado que o do controlador Kp sintonizado por AG.

Controlador Kp	Controlador EHW
246,9287	226,8873

Tabela 5.2: Valor RMS do erro para referência quadrada

Controlador Kp	Controlador EHW
12,0065	11,9916

Tabela 5.3: Valor RMS do erro para referência senoidal

Este capítulo expos os detalhes de implementação dos circuitos controladores Proporcional e Evolutivo para pêndulo amortecido não linear utilizando um dispositivo de lógica reconfigurável FPGA, dentre outros recursos. Também foram expostos, na forma de gráficos e cálculos RMS dos valores de erro, os resultados experimentais obtidos em laboratório. No capítulo 5 serão realizadas as considerações finais desta dissertação.

Capítulo 6

CONCLUSÃO

A aplicação prática de uma técnica recente para o projeto de controladores foi realizada nesse trabalho de mestrado. A proposta de implementar o projeto dos controladores propostos por Campos (Campos (2007))em FPGA e testá-los em condições reais de controle foi realizada, sendo os dados obtidos expostos através de gráficos e cálculos de erro médio quadrado (RMS).

Uma análise comparativa entre dois controladores foi desenvolvida utilizando-se dois sinais de referência (onda quadrada e seno). A mesma planta e condições de testes foram adotadas tanto para o controlador proporcional sintonizado por AG, quanto para o controlador EHW. Dessa forma, com base dos dados obtidos nos testes foi possível observar que o controlador EHW obteve, no geral, um valor de erro inferior ao Kp que implicou em uma diferença de 20,0414 pontos menor para a referência quadrada e 0,0149 ponto menor para a referência senoidal (cálculo do erro médio quadrado). Também foi possível observar através dos gráficos valores de overshoots e erros de regime menos acentuados para o controlador evolutivo, quando aplicado um sinal de referência quadrado ao sistema.

O controle evolutivo demonstra maior habilidade de gerência do sistema devido a sua característica de resposta não-linear obtida através da observação comportamental da planta e uso dessa informação para promover alterações na estrutura interna do controlador adequando-o melhor ao sistema a ser controlado.

Os grandes valores de erro em transições de estado abruptas observados quando um sinal quadrado é utilizado deve-se ao fato de que os controladores são limitados a uma resolução de apenas 4 bits e possuem uma ”visão”do sistema muito limitada, o que implica em respostas de controle também limitadas.

Os dados mostram que apesar de ser uma técnica de projeto de controladores ainda muito recente, a técnica de projeto por hardware evolutivo é promissora e sendo assim caracteriza-se como um campo de pesquisa em ascensão.

6.1 Trabalhos Futuros

Apesar da eficiência comprovada em testes práticos, a técnica ainda deve percorrer um longo caminho até atingir uma maturidade que possibilite seu uso prático, principalmente no geração de controladores digitais mais precisos, já que a técnica EHW tem com o seu principal problema a escalabilidade.

Como trabalhos posteriores é possível citar a implementação e testes em bancada de sistemas mais precisos obtidos também através de técnicas evolutivas. Assim como testes comparativos com controladores PD, PI e PID ou outros controladores convencionais que possuam um tratamento melhor a problemas não lineares de controle. Para a implementação desses controladores, seria necessária a construção de novos blocos lógicos a serem sintetizados no FPGA, com componentes como registradores para armazenamento das 2 amostras de sinal de erro imediatamente anteriores, no caso do geração do controlador PID, por exemplo.

Outra sugestão, também feita por Campos (2007), seria a comparação de controladores obtidos por EHW e controladores obtidos por outras técnicas bio-inspiradas, como redes neurais artificiais, sistemas nebulosos entre outros, de modo que seus comportamentos possam também ser observados em ambientes de controle reais gerados em laboratório.

Apêndice A

Diagramas

Este anexo contém os diagramas dos circuitos digitais do sistema de controle de pêndulo desenvolvidos no ambiente Altera Quartus II e sintetizados em FPGA.

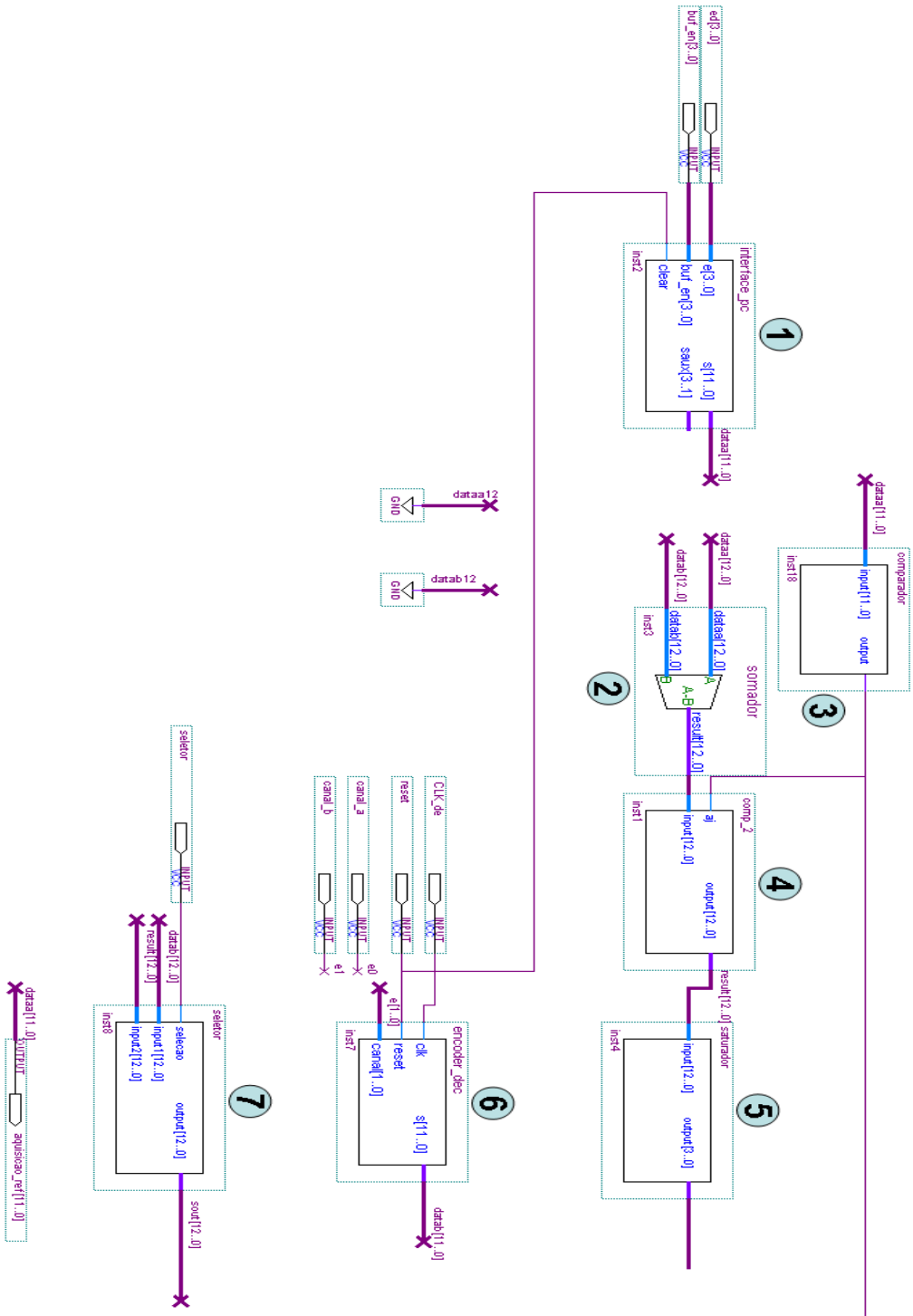

Figura A.1: Diagrama do sistema de controle (parte 1)

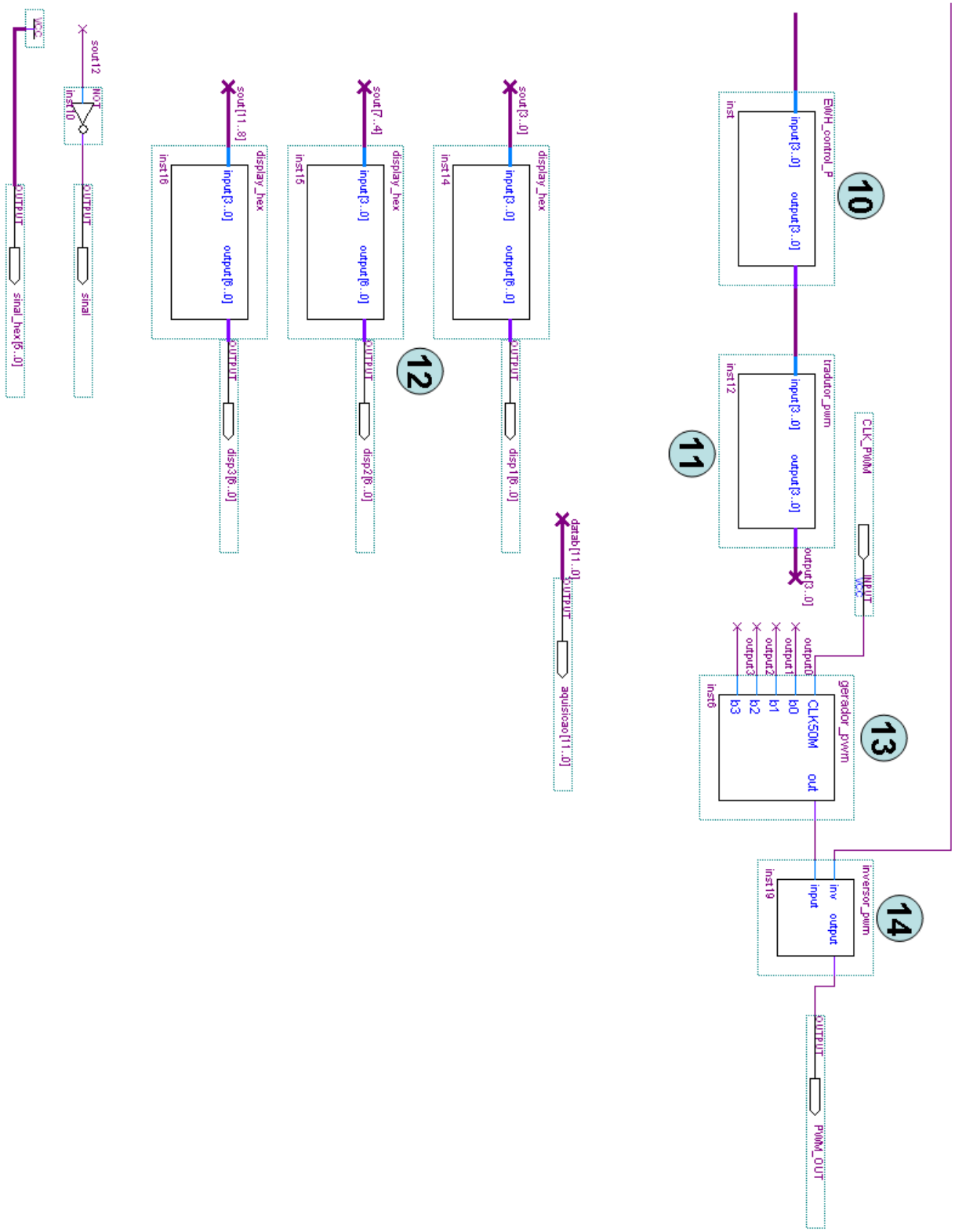

Figura A.2: Diagrama do sistema de controle (parte 2)

Referências Bibliográficas

Altera (2007), *Cyclone II Device Handbook Volume 1*, Altera Corporation.

Altera (2008), 'Página de documentação do ambiente de desenvolvimento Quartus II', URL: `http://www.altera.com/literature/lit-qts.jsp`.

Bäck, T., Fogel, D. B. e Michalewicz, Z. (2000), *Evolutionary Computation 1 - Basic Algoritms and Operators*, Institute of Physics Publishing and Oxford University Press.

Brow, S. e Rose, J. (1996), 'Architecture of FPGAs and CPLDs: A Tutorial', *IEEE Design and Test of Computers* **13**(2), 42–57.

Calazans, N. L. V. (1998), *Projeto Lógico Automatizado de Sistemas Digitais Seqüenciais*, DCC/IM, COPPE Sistemas , NCE, 11a Escola de Computacao, Universidade Federal do Rio de Janeiro.

Campos, T. (2007), Hardware Evolutivo Aplicado a Geração Automática de Controladores para Servo-Mecanismos, Tese de doutorado, Faculdade de Engenharia Elétrica e Computação - UNICAMP.

Campos, T., de Oliveira, J. R. e Jungbeck, M. (2006), 'Hardware Evolutivo Aplicado ao Problema de Controle de Servo-Mecanismos', *XVI Congresso Brasileiro de Automática* .

Coello, C. A., Christiansen, A. D. e e Arturo H. Aguirre (1996), 'Using Genetic Algorithms to Design Combinational Logic Circuits', *ANNIE'96. Intelligent Engineering through Artificial Neural Networks 6* **6**, 391–396.

d'Amore, R. (2005), *VHDL - Descrição e Síntese de circuitos Digitais*, LTC.

Fogel, D. B. (2006), *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 3a. edição, IEEE Press Series on Computational Intelligence.

Gajski, D. e Kuhn, R. H. (1983), 'New VLSI Tools', *IEEE Computer* **16**(12), 11–14.

Green, J. (2006), *A Origem das Espécies - Texto Integral*, Martin Claret.

Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, 1a. edição, University of Michigan Press.

Karnaugh, M. (1953), 'A map method for synthesis of combinational logic circuits', *Transactions of the AIEE, Comunications and Electronics* **72(I)**, 593–599.

Levi, D. (2000), 'Hereboy: A Fast Evolutionary Algorithm', *Evolvable Hardware, 2000. Proceedings. The Second NASA/DoD Workshop on* pp. 17–24.

Mayr, E. (1988), *Toward a New Philosophy of Biology: Observations of an Evolutionist*, Cambridge University Press.

Nicolato, F. (2002), Arquitetura de Hardware para a Extração em Tempo Real de Características de Múltiplos Objetos em Imagens de Vídeo: Classificação de Cores e Localização de Centróides, Dissertação de Mestrado, Faculdade de Engenharia Elétrica e de Computação - UNICAMP.

Nise, N. S. (2004), *Control Systems Engineering*, 4a. edição, Wiley.

Ogata, K. (2006), *Engenharia de Controle Moderno*, 4a. edição, Prentice Hall.

Pazos, F. (2002), *Automação de Sistemas & Robótica*, Axcel Books.

Perry, D. L. (2002), *VHDL - Programming by Example*, 4a. edição, McGraw-Hill.

Pessis-Pasternnak, G., Thom, R., Prigogine, I., Atlan, H., Morin, E., Feyerabend, P., Dupuy, J.-P. e d', B. (1992), *Do Caos à Inteligência Artificial - Quando os Cientistas se Interrogam*, Unesp.

Piennar, R. (1998), 'An egocentric approach to machine intelligence', *Intelligence and Systems, 1998. Proceedings., IEEE International Joint Symposia on* pp. 273–280.

Pomilio, J. A. (2008), 'Notas de aula do curso EE833 eletrônica de potência', URL: `http://www.dsce.fee.unicamp.br/~antenor/ee833.html`.

Sankaran, S. e Haggard, R. L. (2001), 'A convenient methodology for efficient translation of C to VHDL', *System Theory, 2001. Proceedings of the 33rd Southeastern Symposium on* pp. 203–207.

Suzim, A. A. (1998), 'A Cad Frame for VLSI Design', *III Simpósio Brasileiro de Concepção de Circuitos Integrados* .

Zebulum, R. S., Pacheco, M. A. e Vellasco, M. M. (2002), *Evolutionary Eletronics - Automatic Design of Eletronic Circuits and Systems by Genetic Algorithims*, CRC Press.